IOWA STATE UNIVERSITY
Digital Repository

Retrospective Theses and Dissertations

Iowa State University Capstones, Theses and Dissertations

1-1-2005

PIRANHA: an engine for a methodology of detecting covert communication via image-based steganography

Christopher Shaun Pilson
Iowa State University

Follow this and additional works at: https://lib.dr.iastate.edu/rtd

Recommended Citation

Pilson, Christopher Shaun, "PIRANHA: an engine for a methodology of detecting covert communication via image-based steganography" (2005). *Retrospective Theses and Dissertations*. 19216.
https://lib.dr.iastate.edu/rtd/19216

This Thesis is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.

PIRANHA:
An engine for a methodology of detecting covert communication via image-based steganography

by

Christopher Shaun Pilson

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Information Assurance

Program of Study Committee:
Anthony Townsend, Co-major Professor
Brian Mennecke, Co-major Professor
Doug Jacobson
Kevin Scheibe
Jennifer Davidson

Iowa State University

Ames, Iowa

2005

Copyright © Christopher Shaun Pilson, 2005. All rights reserved.

Graduate College
Iowa State University

This is to certify that the master's thesis of

Christopher Shaun Pilson

has met the thesis requirements of Iowa State University

Signatures have been redacted for privacy

TABLE OF CONTENTS

ACKNOWLEDGMENTS ..... **v**

GLOSSARY ..... **vi**

ABSTRACT ..... **vii**

1. INTRODUCTION **1**

1.1. Background 1
1.2. Purpose and Significance of the Research 6
1.3. Scope of Work 8
1.4. Plan of presentation 10

2. LITERATURE REVIEW **12**

2.1. Introduction 12
2.2. Steganography & Steganalysis 13
2.2.1. Steganographic Encoding, Pre-Schola Steganographia Years *13*
2.2.2. Linguistic Steganography after Schola Steganographia *16*
2.2.3. Technical Steganography *18*
2.2.3.1. Microdots 19
2.2.3.2. JPEG Images as a Steganographic Cover 20
2.2.3.3. 4- and 3-LSB Static Encoding in JPEG Images 22
2.2.3.4. Dynamic Encoding in LSB Insertion Systems 24
2.2.3.5. Beyond LSB: Frequency-Domain Encoding 26
2.2.3.6. Advanced Techniques of Steganalysis 29

3. RESEARCH METHODOLOGY **33**

3.1. Introduction 33
3.2. Image File Description 34
3.2.1. JPEG File Specification *34*
3.2.2. Image Domains *35*
3.3. Transforms and Signal Processing Used 37
3.3.1. Discrete Cosine Transform (DCT) *37*
3.3.2. Histogram (grayscale values) *40*
3.4. Equipment 41
3.4.1. Personal Computer Used *41*
3.4.2. Conversion, Analysis, and Modeling Software Used *41*
3.5. Image Preparation 42
3.5.1. Image Description *43*
3.5.2. Image Size *44*
3.5.3. Image Context *44*
3.5.4. Image Selection *45*
3.5.5. Image Warehousing, Format, and Storage *46*
3.5.6. Image Naming Convention (and why this is important) *46*
3.6. Model-Building and Validation 47
3.6.1. High-Level Description and Process Flow *47*
3.6.2. Artificial Neural Networks as a Methodology *53*

4. RESULTS **55**

4.1. System Output and Analysis 55

4.1.1. Runtime *55*
4.1.2. ANN Modeling Accuracy *56*
4.2. System Uniqueness 58

5. CONCLUSION 60

5.1. Summary 60
5.2. Discussion of Findings 61
5.3. Limitations of the study & Future Work 62
5.3.1. Pluggable modules *62*
5.3.2. Open system *63*
5.3.3. Flexible architecture *63*
5.3.4. Hardware Implementation *64*

REFERENCES 65

APPENDIX A – LISTING OF STEG_NLSB.M 69

APPENDIX B – LISTING OF STEGHIDE.M 71

APPENDIX C – LISTING OF ANNPREPROCESS.M 73

APPENDIX D – VISUAL INSPECTION OF INFECTED FILES 76

APPENDIX E – ARTIFICIAL NEURAL NETWORK CLASSIFICATION ON AUTO- AND XENO-DATA 78

APPENDIX F – INSERTION RUNTIME VS. TYPE OF IMAGE DOWNGRADE 80

APPENDIX G – TRUE, FALSE, AND MISSED POSITIVES DETECTED 81

APPENDIX H – SYSTEM BUILD TIME AND ACCURACY VS. LSB INSERTION 82

APPENDIX I – GRAYSCALE PHOTO PLATES FOR IMAGES UTILIZED 83

ACKNOWLEDGMENTS

I want to take a moment to reflect and thank the many people who helped make this thesis a reality. I would like to start by thanking my parents for instilling in me the desire to learn and discover, as well as occasionally suffering through hearing about the strange world of academia with enough patience and objectivity to see me through. Without their encouragement and support for my move across the country to attend Iowa State University to pursue a graduate degree, this thesis would not exist. My wife deserves thanks for being supportive and intermittently interrupting the work and demanding that I sleep. I also would like to thank my committee chair, Dr. Anthony Townsend, for initially helping me adjust to the rigors and peculiarities of graduate school, and later (and currently!) for always seeming to have time to hear "yet another" eccentric suggestion that I had for future research directions and topics – including this thesis. It is this freedom and collaboration that has made this journey achievable through the notion that it would somehow be accomplished and, moreover, that I was up to the challenge. I have never heard the phrase "you can't" from him, and deeply appreciate the license to explore that this gives. I would also like to thank Dr. Brian Mennecke for his consideration and openness, as well as his continued support and collaboration. I appreciate both Dr. Scheibe and Dr. Davidson's input into the thesis by providing insight into and refinement of the problem space, as well as Dr. Davidson's observation that the ideas presented had merit. I am grateful to Juan Melendez for his continued good humor and friendship through the process, and appreciated his input (including permission to use some vacation pictures) in the writing of the thesis. I also want to acknowledge Alf Green and his help in getting a toe-hold into the programming of Matlab, as well as his interest in the topic. I am also grateful to the numerous people in my life who understood what an involved process this has been, and who also suffered through the inexorably boring details of another person's thesis topic housed in a specialized and difficult knowledge domain. Finally, I would like to thank you, the reader. Too often, it goes unacknowledged that these works have an audience – but without an audience to read it, any such work would be worthless. I have taken pains to present the information in a clear, concise, and useful manner. Hopefully the text that follows will be an interesting and engaging experience.

GLOSSARY

DCT – Discrete Cosine Transform. A transform which presents only the real component of an FFT (Fast Fourier Transform).

JPEG – Joint Photographic Experts Group. An image standard that utilizes the DCT transform and a quantizer to achieve image compression.

LSB – Least Significant Bit. Bit zero, the bit of a binary number giving the number of ones, the last or rightmost bit when the number is written in the usual way. In steganography, this refers to a method of image embedding that removes the least significant bits from the cover image, and injects covert information in its place. Traditionally, 3- and 4-LSB models have enjoyed widespread use and popularity, meaning that for every (8-bit) pixel, up to 3/8 or ½ of the information given at that point may in fact belong to the covert data.

PIRANHA – Acronym. Short for **PI**cture **R**ecognizer **A**nd I**N**jection **H**ardware **A**ppliance. A system which will house lightweight components evolved from those within this thesis as its engine, and will work with the methodology of analysis presented here to build an emergent picture classification system.

Stego-Image – An image file which contains steganographically embedded data. Also called a "cover image" or "cover carrier".

ABSTRACT

In current cutting-edge steganalysis research, model-building and machine learning has been utilized to detect steganography. However, these models are computationally and cognitively cumbersome, and are specifically and exactly targeted to attack one and only one type of steganography. The model built and utilized in this thesis has shown capability in detecting a class or family of steganography, while also demonstrating that it is viable to construct a minimalist model for steganalysis. The notion of detecting steganographic primitives or families is one that has not been discussed in literature, and would serve well as a "first-pass" steganographic detection methodology. The model built here serves this end well, and it must be kept in mind that the model presented is posited to work as a front-end broad-pass filter for some of the more computationally advanced and directed stganalytic algorithms currently in use.

This thesis attempts to convey a view of steganography and steganalysis in a manner more utilitarian and immediately useful to everyday scenarios. This is vastly different from a good many publications that treat the topic as one relegated only to cloak-and-dagger information passing. The subsequent view of steganography as primarily a communications tool useable by petty information brokers and the like directs the text and helps ensure that the notion of steganography as a "digital dead-drop box" is abandoned in favor of a more grounded approach. As such, the model presented underperforms specialized models that have been presented in current literature, but also makes use of a large image sample space (747 images) as well as images that are contextually diverse and representative of those seen in wide use.

In future applications by either law-enforcement or corporate officials, it is hoped that the model presented in this thesis can aid in rapid and targeted responses without causing undue strain upon an eventual human operator. As such, a design constraint that was utilized for this research favored a False Negative as opposed to a False Positive – this methodology helps to ensure that, in the event of an alert, it is worthwhile to apply a more directed attack against the flagged image.

1. INTRODUCTION

1.1. Background

With the widespread adoption of the Internet as a public and private communication intermediary, message privacy has fallen under increasing scrutiny over the past decade. The inherent power of the Internet with respect to communication is also its downfall – as the Internet is a network that is both public and ubiquitously accessible from many network edges, communications within the network are inherently open to inspection by third parties. This concept was born out of Gus Simmons' thought-experiment of 1983 (Simmons, 1984), which pitted two prisoners (*Alice* and *Bob*) the task of sending messages past an observer – appropriately referred to as a *Warden*. The goal was for Alice and Bob to communicate in a manner that would not arouse the suspicion of the Warden, through the concept of a *covert channel*. As the modern Internet has grown out of a fully open and public infrastructure (first ARPANET, then a blended FIDONet/NSFNet architecture), this analogue holds today – Alice and Bob may represent any two private parties, and the Warden that looks over their shoulders has now expanded to include every node that is within a collision domain residing along the public channel between Alice and Bob. This clearly puts private communications at a disadvantage, but as personal computing power has increased, so too has the corresponding complexity of the encryption options open to private individuals using the Internet to pass messages. While it may potentially be argued that both current implemented and widely-used encryption standards (DES, 3DES), and largely under-utilized encryption methods (AES) more than outstrip present-day machinery and demands, encryption itself does not solve all problems

involved within in-group communication across a public network infrastructure as the Internet. Encryption, at its core, attempts to mask a message by passing the plaintext through a one-way function, such that it is intractably difficult to "decode" the resulting ciphertext via the same mechanism that was used in encryption (Bishop, 2003). This naturally infers, then, that the security of such a system *does not reside* within the secrecy of the algorithm used (although a notable exception comes with the Black Box notion within the DES algorithm). However, with the ability to build affordable, scalable clusters and grids using off-the-shelf commercial technology that has come about in the past ten years, and the implementation of quantum computing perpetually "just around the corner", encryption algorithms that have historically been considered intractably difficult have now become attackable by commonplace hardware. One mechanism that marks the defeat of an encryption algorithm is the ability to create a collision, such as was found in both MD5[1] (Rivest, 1992) and SHA-1[2] (N. S. Agency, 1995) security algorithms recently. Even with the assumption that the future will bring about the creation of new and more robust encryption algorithms, the fundamental problem present within the dependence upon encryption as a mechanism to secure communications still exists – the issue is one of *participant discovery*. For instance, if group *G* is set up such that *G{a,b...z}* (where *a,b...z* are members of group *G*), if *a* sends a message to *b*, who then relays the message on to members *c, d,* and *e*, then it is very possible for a Warden to achieve, given enough traffic, full disclosure of the membership of *G*. With the technological advances

[1] CertainKey will award its $10,000 bounty (Inc., 2004) to a trio of Chinese researchers [the same group that is mentioned in the footnote below for compromising SHA-1] for the feat. MD5 was broken in February of 2004.
[2] This is just starting to unfold as of February 18, 2005 – the short of it is that a trio of Chinese researchers [Xuejia Lai, Xaioyun Wang, and Hongbo Yu of the Dept. of Computer Science and Engineering at Shanghai Jiaotong University in Shanghai, China] devised a collision attack for SHA-1 that is about 2000 times faster than a brute force attack; see Bruce Schneier's public web log (Schneier, 2005) for more details.

discussed above, and the ubiquity of powerful computers, the desire for Agents within *G* to be able to secretly send covert messages to each other is increasingly more poignant.

Traditional one-to-one communications between Agents within group *G* have given way to one-to-many and even many-to-many communiqués in recent years with the proliferation of network-attached edges of the Internet, and the corresponding lowering of the "cost of admission" to Internet participation – in both monetary cost as well as through pre-built network applications. Electronic mail has experienced a decline in usefulness for message dissemination, both logistically as free and open email proxies have diminished and also through the growing popularity and use of larger-scale communications such as web logging ("blogging") and message boards. These two solutions both avoid exposing endpoints of the communication and also enable many-to-many communications to take place with relative ease. As such, the communication is able to take place through a virtual "cloud" rather than a one-to-one pipeline between participants, and therefore becomes much more difficult to uncover. With this foundation in place, it becomes apparent that the next generation of data encryption and security will not only rely upon more complex ciphers, but must also make use of the currently underutilized concept of masking the message within an innocuous carrier. This masking of the target with spurious or innocuous information has the end result of making the separation of signal from noise increasingly more difficult and devastatingly more time-consuming from the perspective of a Warden operating within the network.

The next battlefield of information security clearly must be focused on the issue of determining the *scope* of information dissemination. As has been mentioned earlier, it is convenient to believe that *any* encryption algorithm is at risk of exposure. Therefore, given enough computing power and/or message samples, algorithms may be reverse-engineered

through the use of data collisions and statistical measures. Putting this aside, even fiendishly and *impossibly* difficult algorithms applied to traditional communications will still uncover involved parties in the communication. Therefore, the next generation of communication across public channels (such as the Internet) must meet two disparate ends: the communication must be *ultimately and widely accessible* while remaining *secure and covert.* There exists a largely under-utilized methodology of presentation that meets these ends – s*teganography.* Steganography holds that data may be inserted into an innocuous carrier stream – for the purposes here in this paper, this carrier is an image produced in coordination with the JPEG file specification – and transforms results in the creation of a *stego-image*, an image indistinguishable from the original image upon casual inspection by a third party. The net result of this transformation fits *perfectly* into current covert communication requirements as the data becomes both hidden in "noise" and also may be encrypted via any method available to the parties involved in the communication[3], thus protecting the message from direct inspection should its presence be uncovered.

Even through terms such as "covert message", "covert channel", and "prisoners' dilemma" all sound quite cloak-and-dagger, there exist a good many scenarios in which disclosure of membership within a communicating group is far from the back-rooms of three-letter governmental agencies. For instance, hoboes' writing of the 1930s has recently reappeared in the form of "warchalking", which makes use of cryptic symbology to convey information regarding wireless access points covertly (Thomas, 2004). A more business-oriented example might be found in a transaction where it is potentially as or even more

[3] As steganography makes no underlying assumptions to the format of the source, it is be possible to utilize any and all cryptographic methods in sending a stego-image – for instance, pairing stego-imagery with PKI will result in a message that is covert, secure, *and* can verify the sender through use of message signing.

important to protect the *members* of the discussion than the discussion itself – details of a merger taking place between two companies, discussed by their respective principals, would certainly present a non cloak-and-dagger and above-the-board use for steganography. However, it is important to note that this particular application is given here only as a counterpoint to present steganography in a typically neglected light – this thesis treats subsequent mentions of steganography and steganographic embedding as a form of communication that is unwanted on the network, dangerous, and potentially illicit.

Granted, in the face of the above notion that not *all* use for covert communications is cloak-and-dagger, it provides an obvious advantage for a group to remain undetected for as long as possible when communicating messages in the presence of a Warden. As such, and as groups become more globally diverse and far-flung, ascertaining *group membership* and building social network maps of these groups is becoming more imminently important than necessarily discovering just *what* the agents within the groups are saying to each other. A good deal of research in communications security has been spent in conducting an "arms race" of sorts through building increasingly more mathematically intractable one-way functions for encryption. Conversely, faster cryptographic algorithms with which to defeat these have also been built; thus, it is more productive to instead gain an insight into *which* messages contain spurious information for the purpose of finding the involved parties *of* the communication. However, even with the notion that steganographic communication is not *only* to be relegated to cloak-and-dagger operations, it is typically a good mental shortcut to extend research in this field to include complete utilization by governments and subversives alike. In doing so, the trap of seeing steganography only in the light of what Bruce Schneier might dub a system to

"fool your kid sister" is avoided, as steganography is most assuredly *the other kind* (Schneier, 1996) of secret communication methodology.

Given the importance and urgency of the above paragraph with respect to global affairs, this thesis will introduce a machine-learning engine – utilizing an Artificial Neural Network – that will potentially aid in the discovery of covert communication endpoints by determining if a presented image is *suspect*, which will imply that the image is a probable stego-image, or if the image is instead *benign*, meaning that the image appears to not be a covert channel for information.

1.2. Purpose and Significance of the Research

Steganalysis, as Chandramouli et al. note, has two primary approaches: one may build a steganalytical attack against *one* known steganographic technique, *or* one may build an attack that is able to detect the existence of steganography, but will be inherently unable to determine *which* steganographic method is employed (Chandramouli, Kharrazi, & Memon, 2003). This thesis, however, takes a *blended approach* as of yet unseen in the research stream. This is an important "next step" for the steganalysis to take, as new steganographic techniques are being developed continuously, and it is simply not granular enough to give an examiner a simple "yes" or "no" answer to the question "*is steganography present within this image?*". Rather, an examiner must be able to look at an analysis which will be able to present a confidence rating from several different *styles* of steganographic embedding *in addition to* a unified model. In this fashion, current techniques will be captured (e.g. "this new data conforms well with an LSB-insertion analysis") as well as future steganographic techniques.

Unlike some other steganographic and image techniques that are aided through use of a difference image (for an example, see (Mendoza, 1999)), the method presented in this thesis for steganalysis make *no* assumption about the existence of a difference image, thus enabling the engine to view entirely new images and determine if the image is suspect or not. Indeed, since recent literature regarding steganography makes both explicit (Lou & Liu, submitted) as well as implicit (Lou & Liu, 2002) mention of the "Common Cover Carrier Attack", in which acquisition of a *difference image* (e.g. a common cover image that carries the embedded covert information) trivializes detection of covert information, this thesis will take the position that *no* such Common Cover Carrier Attack may be carried out – in other words, the method presented in this thesis has *zero a-priori* knowledge with respect to the content of the images as they are processed. Section 2.2.3.3 carries a discussion on the weakness of steganographic and watermarking techniques when difference images are utilized as expressed by Petitcolas et al. (Petitcolas, Anderson, & Kuhn, 1998).

It is vital to come back to a point that has, thus far, been largely implied: the type of model-building present in this thesis has two primary goals. The first, of course, is to predict and classify images as either "stego-imagery" or "benign". A sub-goal of this, as development of the PIRANHA engine will continue to grow beyond the scope of this thesis, is to err on the side of *missing* true positives, while instead focusing upon providing a good probability of a *reported* positive being an *actual* positive. In short, the penalty for mis-classifying a benign image as a stego-image is higher than missing a stego-image by calling it benign. This is an important research consideration to make note of. The second main goal, which is unique to this thesis, is to find a *minimal* model that will yield adequate classification results. This is a worthwhile venture, as the PIRANHA engine attempts to provide for a much more useable and

"real-world" model than is seen in publication. For instance, in the example about information brokering and leakage by a company insider through steganography, it is reasonable to assume that the insider will not build a highly complex second-order model of the image file and then dynamically insert the covert information within it accordingly; rather, it is probable that they will either use a pre-packaged tool (such as *Steghide*), or write a quick program to achieve static n-LSB insertions as is discussed in Kurak (Kurak & McHugh, 1992).

The above is viable, interesting, and significant in its simplicity and targeted nature – in effect, many individuals using steganography to embed information are likely *not* going to be image-processing experts. Hence, a model which will be maximally effective will do two things: it will look at a gargantuan body of images and be able to do so in a reasonable time, and will strive to reduce the workload on an individual operator wanting to work to extract data or further analyze the reported stego-image. A model that tries to minimize computational time and cognitive and computational complexity is parsimonious with these approaches in its simplicity of operation – and, given enough individual modules making decisions, it is possible to create an engine that may well outperform any single monolithic engine through its emergent properties.

1.3. Scope of Work

This thesis is concerned with the following primary goal: the creation of an engine that will detect both current and future forms of technical steganography. This thesis looks at steganography in *one and only one* context with regard to analysis: *the embedding of covert information in JPEG image files.* This kind of a bounding condition is necessary, given the plethora of steganographic carrier media (refer to Section 2.2, below, for a detailed overview and analysis) in current use. In designing the PIRANHA engine, however, great pains were

taken to avoid a special-purpose system that could *only* be used for this purpose. There is, therefore, no reason why the method presented in Section 3 could not be expanded to detect messages within other kinds of carrier media (e.g. BMP files, which embed covert bits physically in the spatial-domain of the cover image; GIF files, which make use of palette shuffling techniques [common computer programs to achieve this include *EzStego* (Machado, 1996), *Gifshuffle* (Kwan, 2003), and *Hide and Seek* (Moroney, 1996)] to embed 1675 [$log_2(256!)$] bits (Kwan, 2003) of covert data into the file). There is also no reason why the method presented cannot, by extension, also detect different *methods* of steganography in either JPEG carriers or other media.

As the thesis progresses, it is important to take note that, given the content, topics covered will naturally include a defense of the investigative methodology utilized (Artificial Neural Networks), as well as either butting against or encroaching upon topics within watermarking and document distribution and cryptography. This takes place intentionally, and is a necessary part of providing full coverage into steganography. Figure 1.1, adapted from Bauer (Bauer, 1997), provides an overview of where steganography fits into the more global picture of information embedding as well as the inclusion of watermarking as a *specific type* of steganography.

As the suggested communication strategy makes use of steganography over widely used public channels, the classification engine concerns tries to accomplish the following goals:

- Repeatable and reliable classification of an image as either *suspect* or *benign*,
- Output of image analysis in a common format, such as a comma-separated file.
- Compatibility and interoperability with commercially available software.

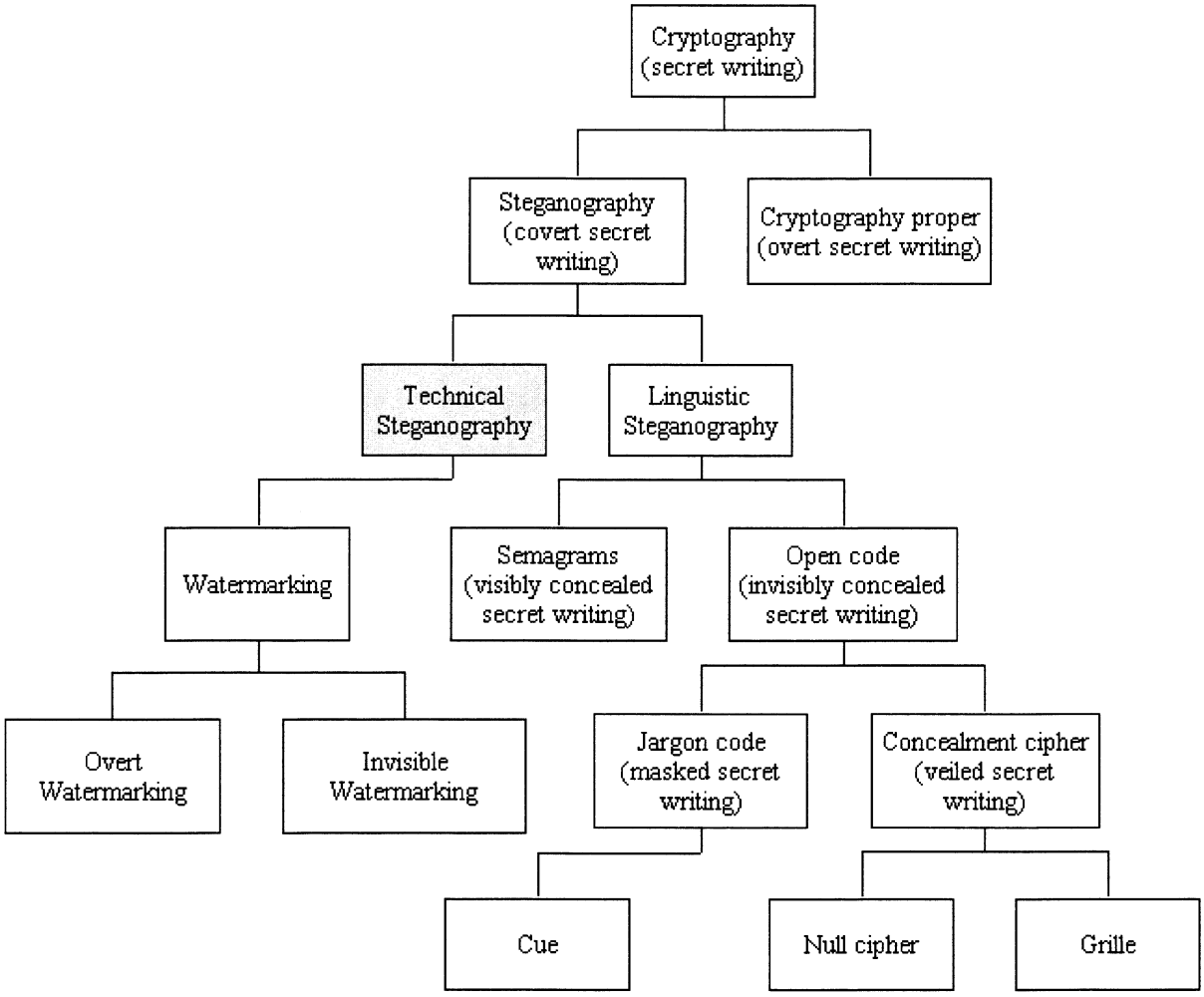


Figure 1.1 - Steganography *in situ* with other information hiding schemes (adapted from (Bauer, 1997))

1.4. Plan of presentation

Chapter 2 will present the literature review for steganography. Additionally, the "arms race" that has been touched upon in the introduction between steganography and steganalysis will be expanded and analyzed.

Chapter 3 will present the research methodology, and cover topics endemic to the research problem. Such topics include discussions regarding the JPEG file standard, image processing, equipment used in the research, data preparation and warehousing, and model building.

Chapter 4 will cover results achieved using the methodology described in chapter 3. Moreover, this chapter will also provide a feedback loop for the equipment used and methods developed in chapter 3.

Chapter 5 will engage the reader in a discussion of the research, and present a discussion of the findings from chapter 4. Furthermore, chapter 5 will look at the limitations of the current study and make suggestions on the improvement and extension of the PIRANHA engine.

2. LITERATURE REVIEW

2.1. Introduction

The art of *steganography* – literally "covered writing" – has likely existed in one form or another for as long as humans have been inclined to keep written secrets. A well-known anecdote from Herodotus includes Histiæus shaving the head of a trusted slave and tattooing a message upon the scalp (Herodotus). The hair was then allowed to regrow, at which time the slave made the journey to the message recipient. Were the slave-cum-messenger to be detained en route, the secret was typically kept safe under the cover of the man's hair. Indeed, this low-tech approach to steganography was still in use by spies in the early portion of the 20th century (B. Newman, 1940). However, for the purposes of this thesis, Gaspar Schott's 1680 work, *Schola Steganographica* (Schott, 1680), will suffice as marking the first point at which discourse at length was formally undertaken and published involving steganography[4]. While I was unable to acquire a copy of the work for review, its inclusion here marks an important realization: steganography is by no means a new – or even young – discipline, and has enjoyed centuries of use and protection.

What *is* changing significantly in our present epoch, however, is the art of steganalysis. Steganalysis may be thought of with regard to steganography as decryption is to encryption: it provides a means of *discovering* the presence of steganography. However, it must be noted that the same mathematical and technological advances that are furthering steganalysis are also

[4] Trithemius' earlier work, *Steganographia* (Trithemius, 1621), is not considered in this thesis because it presents steganography simply and principally only as a vehicle for the preservation of occult writing and teaching. Even though Schott's work extended on this initial treatise, it is less focused upon the occult and more upon the examination of information embedding, as is seen in its encoding of an alphabet within the notes of *Ava Maria*.

means to more effectively and efficiently hide the message in "noise", and the techniques present on both sides of the problem are gaining momentum. As will unfold below, this environment leads to a sort of "arms race" between two factions: researchers *creating* more effective algorithms for data embedding, and those *unraveling* the work of their peers, and striving to expose innocuous-appearing carrier signals as instead a wrapper around a secret message.

2.2. Steganography & Steganalysis

Steganography, as has been mentioned above, occurs when *any* signal purposefully inhabits cover noise to mask its presence. It naturally follows from this that there can be many instantiations of steganography – indeed, there may be as many instantiations of steganography as there are types of carriers. This thesis will be constrained to the discovery of steganographic embedding within computer image files – namely, JPEG images – but, for completeness, will provide a high-level coverage of alternate carriers of steganographically embedded information. The next few sections will cover literature pertinent to exploring steganography first in the absence of computer images completely, and then in the absence of properties unique to JPEG images, and then finally steganography involving feature extraction with features unique to JPEG files. After these steganographic methods have been examined, this thesis will provide an in-depth analysis of the "arms race" that has earmarked the development of progressively ever more intricate and intractably difficult practice of steganography.

2.2.1. Steganographic Encoding, Pre-*Schola Steganographia* Years

Earlier, it was mentioned that the "birth" of measured discourse upon steganography came at the hands of Schott. However, the realization must be made that this is *not* to say that other methods did not come before *Schola Steganographia.* From Herodotus' description of Demeratus' clever ruse involving a "clean" wax tablet to warn Sparta of the Persian invasion to Æneas the Tactician's writing of messages hidden on the soles of the messengers' feet (Tacticus, 1990), it is clear that Schott is not the pioneer of steganography, but rather compiled most of his tome from information available in 1680. Both linguistic and artistic steganography (an offshoot mentioned in this section alone) existed before Schott's work, with a predominance placed upon the former form of steganography. As the concept is seldom solidified, a quick word is needed to provide an exemplar for the concept of artistic steganography.

Artistic steganography is a form of covert communication via traditional artistry – painting is the easiest medium to imagine. Shö's work the *Vexierbild* typifies artistic steganography, in that extreme-angle imagery was used within the painting, such that the content shifted between that of an abstract landscape to portraits of kings if viewed from the correct vantage point.

Linguistic steganography, by comparison, works to "codify" a hidden message in an innocuous cover text. Typically, this end has been met through use of either the traditional embedding a "pattern" of letter use within the text itself (e.g. "every third letter of every word contains the covert message"), or the *spacing* of either inter-character or intra-line layout with the advent and wide use of automated publishing tools such as the printing press. Giovanni Boccaccio's *Amorosa visione* and the *Hypnerotomachia Poliphili* (Anonymous, 1499) serve as basic examples of linguistic steganography; *Hypnerotomachia Poliphili* contains a pointed

message that was physically spelled out in the first letter of the 38 chapters[5]. Francis Bacon, in book IV of his 1623 Latin edition of "Advancement of Learning" (Bacon, 1623), divulges a cipher that he claims to have invented – the Biliteral Cipher[6]. Additionally, the actual embedding process into cover text takes place by making very slight alterations in the physical structure of the marked letters themselves. This slight alteration was then used to reproduce a "key" wherein five characters[7] were utilized to compose one single decrypted letter. This principle of letter-by-letter encoding and decoding through grouping binary digits (bits) together later went on to be used and standardized most notably by Samuel F. B. Morse in 1844, and then Emile Baudot in 1875 (through use of a five-bit system) for the transmission of messages. Morse code, by its design, did not lend itself well to automation; Baudot code (Tampa Bay Interactive, 1998a) went on to enjoy further success as a representation and communication strategy in automation and machinery, culminating in its use in the teletypewriter (Tampa Bay Interactive, 1998b). The embedding methods covered thus far have been periodic and cyclical in nature, meaning that once a single instance of linguistic steganography has been discovered, the pattern itself may be played out over the entire body of the text to divulge more of the original message text, leading to a complete recovery of the previously hidden message. However, even prior to *Schola Steganographia*, rudimentary groundwork had already been laid to overcome this – ancient China used a method of "masking" a blank sheet of paper with a template with holes placed at random intervals along

---

[5] *Brother Francesco Colonna passionately loves Polia* is the message.

[6] Incidentally, it is through finding evidence of the Biliteral Cipher in the works of Shakespeare that some scholars have reached the conclusion that perhaps Bacon held the pen behind the works. A discussion on this is beyond the scope of this thesis, but is an interesting segue nonetheless.

[7] Each plaintext character was comprised of five "cover" characters, made up of a sequence of either *a* or *b* characters. In short, this utilized a binary system that ranged from *aaaaa* to represent "A" to *babbb* for "Z". It is, then, identical to the Baudot code *but for the fact* that Bacon's Biliteral Cipher sequentially incremented both cipher- and plain-text representations: 00000=aaaaa="A", and 00001=aaaab="B". Baudot code, while still made up of 5 bits, is non-sequential.

its body. The covert message would be written into these spaces, and with the mask removed and cover text added, the message was secure to all but the recipient, who had a copy of the "key"[8]. This method of covert communication was later formalized by the Italian mathematician Cardan in the early 16th century, resulting in what came to be called the "Cardan Grille" method. However, early steganography had the same earmark: individuals involved in the message had either key management (in the punched-template example) or the security of a covert channel (in the altered-type examples given) to contend with. It will be shown that later, this basic choice and weakness was removed, and replaced by intractably difficult methods of steganography.

2.2.2. Linguistic Steganography after *Schola Steganographia*

Recall from previous discussion that linguistic steganography hides a covert message within a larger text carrier. After Schott's *Schola Steganographia*, steganography took a decidedly more clandestine turn in linguistic steganography by altering *line spacing* rather than individual characters. This made the discovery of a hidden message much more difficult than using marked letters – for instance, even if the marked letters were used in a highly complex encoding scheme, the mere *presence* of such letters making themselves known would eliminate the key concept of steganography, and would reduce the problem to a rote cryptography exercise. Current-day systems make use of both line-shifting and word- or character-shifting techniques[9], and operate within extremely tight tolerances; to do this, they must establish a

---

[8] It is interesting to note that some *modern* cipher systems (DES, 3DES) represent the same notion here – that of a shared *symmetric key* held by all who wish to join in the conversation. In fact, many of the problems that must have plagued ancient Chinese and Italian participants are still faced today, with key management and key security at the forefront.

[9] While a variety of these systems make use of line- and character-shifting for document marking purposes, these same techniques lend themselves to the *encoding* of information rather well.

protocol of sorts that governs the document's appearance and layout. For instance, if a line is to be shifted up (binary 1) or down (binary 0), it must come between two *un-shifted* lines. This works to ensure a "control" scenario in which nuances of document imaging and copying may be accounted for. The actual line of text is shifted only 1/150 of an inch (Low, Maxemchuk, Brassil, & O'Gorman, 1995), which is impervious to typical visual inspection. This method chooses 1/150 of an inch to ensure robustness; an article by the same authors (Brassil, Low, Maxemchuk, & O'Gorman, 1995) demonstrates that 1/300 of an inch is a largely viable and wholly undetectable measure. The line- or word-shifting method of linguistic steganography is successful against the human visual system because 1/300 of an inch is a resolution much finer than a human observer may perceive. However, the use of a computer has rendered this approach unsuitable for the transmission of highly sensitive messages. As is suggested in *Electronic Marking and Identification Techniques to Discourage Document Copying* (Brassil et al., 1995), all a user has to do to detect the presence of such marking within a document can be as trivial as magnifying the image and physically counting pixels between lines, or as complex as invoking a pattern recognition tool such as the "horizontal projection profile", which will detect any shifted lines of text. Additionally, computer algorithms such as centroid detection may be invoked (Low et al., 1995) to detect both line and word shifting. As such, systems that make use of line- and word-shifting have been largely relegated to the domain of content distribution discovery rather than enjoying further use as a means by which embedded information may be conveyed. Other linguistic steganography systems, too, have been handily defeated through use of computer algorithms – a clear example comes in David Kahn's *The Codebreakers* (Kahn, 1967), in which a message was sent from a German Spy in World War II. The cover message reads innocently enough:

> Apparently neutral's protest is thoroughly discounted and ignored. Isman hard hit. Blockade issue affects pretext for embargo on by products, ejecting suets and vegetable oils.

This message, though, acts as a cover for the covert message of "Pershing sails from NY June 1" when a viewer takes only the second letter of each word to form the message. This approach, as it is programmatically easy to implement and iterative, is trivial to defeat. A more robust (and modern) example comes from Maher's *Texto* program (Maher, 1997), which takes plaintext and embeds it into a cover comprised of nonsensical pseudo-poetry. While this is indeed cryptic, it suffers greatly from even subtle alteration attacks (e.g. such as may be experienced with an Active Warden), as long as the attacks are launched "appropriately". Additionally, *Texto* creates poetry which is, by the author's own admission, "*really* repetitive" (emphasis his; from the README file within the *Texto* package), as well as creating cover text that is 10 times larger than the original text. A non-invasive method of surreptitious data insertion within a text document cover comes from Matthew Kwan's *Snow*, which hides information at the end of each line through the addition of between 0 and 7 spaces, thus enabling 3 bits per line to be hidden (Kwan, 1998). Clearly, this kind of embedding mechanism is unsuited today's environment, which typically necessitates a higher-capacity storage scheme. On this note, then, begins the fall of linguistic steganography and corresponding meteoric rise of the second principal type of steganography – technical steganography.

2.2.3. Technical Steganography

Technical steganography is distinguished from linguistic steganography through the utilization of a non-text carrier, and through the use of specialized tools or toolkits. Figure 1.1, from Section 1.3, highlights technical steganography's "fit" into a broader classification of

methods that serve to covertly embed information within an overt carrier signal. Two broad breakdowns that will be provided here include use of microdot technology, and image injection techniques from static 4-LSB insertion methods to fully dynamic systems that scale with the image itself.

2.2.3.1. Microdots

Microdots are printed messages that make use of their minute size to conceal message contents (Kahn, 1967). Many sources inappropriately credit "the famous Professor Zapp, [with inventing] the micro-dot process, at the Technical High School in Dresden" (Hoover, 1946) – indeed, the proceeding quote comes from a *Reader's Digest* article released by J. Edgar Hoover of the U.S. Federal Bureau of Investigation. While *Walter* Zapp, a Latvian engineer, created the Minox camera (C. I. Agency), he did not create the process of microdot creation itself. Instead, this honor correctly belongs to Emanuel Goldberg, who handed out images, each dubbed a *Mikrat nach Goldberg* – a "Goldberg-style microdot" – with a resolution exceeding 1 micron at the Sixth International Congress of Photography in Paris in 1925 (Buckland, 2002). Essentially, microdots represent an extreme level of image reduction onto film, which typically (during and after World War II) was then affixed under postage stamps or onto individual characters (such as a period – "."(Hoover, 1946)) of correspondence. Microdots are typically viewed as either an "old" technology, as in the case of classic World War II tales, or as an anti-counterfeiting measure; yet the simple microdot is enjoying a comeback. In fact, U.S. patent 6,312,911 was awarded for a recent development of using DNA-based microdots message embedding purposes (Clelland, Risca, & Bancroft, 1999)[10]. Additionally, in early August of

[10] More recent advances have come with the realization that this same technology can be used for watermarking, which is a specialized kind of steganography, as well as authentication (Chartrand, 2002).

2004, *Wired* carried an article (Leahy, 2004) in which microdots are revealed to be used in automobile applications as a form of part tagging, which makes prosecution of reselling stolen car parts a successful prospect. Clearly, microdots have their purpose as covert carriers of information, but their use for intelligence-gathering or covert message passing has largely fallen out of common use with the decline of postal communications and corresponding rise of electronic communications.

2.2.3.2. JPEG Images as a Steganographic Cover

In any digital image representation format, there exists some "slack" amount of space, which may be utilized for hiding information without noticeably degrading the image itself. Different steganography tools will provide different embedding capabilities with respect to both utilization and evasion of detection. In his 1991 article (Wallace, 1991), Gregory Wallace presents the JPEG file format as a potential standard in digital image compression for continuous-tone grayscale and color images. An important note to keep in mind, as more on the file specification and steganographic embedding techniques are discussed, is that JPEG, through quantization, is a *lossy* compression[11] routine. It *is* so by virtue that quantization itself is a many-to-one mapping. The JPEG file specification utilizes a *block-based* encoding scheme, which partitions the source image into a series of 8x8 pixel blocks, and encodes these blocks through use of a DCT algorithm (sometimes referred to as the FDCT, or *Forward*-DCT, algorithm). For lossy images, the 64 (8x8) DCT coefficients are next passed through a quantizer, which discards information that is not visually significant. Finally, the blocks are passed through a Huffman or other such entropy encoder, which is able to provide lossless

[11] The JPEG Still Picture Compression Standard (Wallace, 1991), however, *does allow* for JPEGs to utilize *lossless* encoding. Typical JPEG files, though, are so encoded specifically due to the desirability of smaller filesizes while maintaining a modicum of image fidelity. Therefore, for the purposes of this thesis, *all* JPEG files will be assumed to be compressed using the lossy encoding algorithm laid out.

compression by encoding the DCT coefficients based upon their statistical characteristics. Please refer to Figure 2.1 for a graphical overview of this process. The JPEG file format marked an important development for computing and message embedding alike, given its tractability of computation (every visual element is governed by membership in an 8x8 aggregated block that is reasonable in its processing time), and also its multiple representation of visual elements. Considering that the quantization process takes *an approximation* from its source to present a single visual element, the realization may be made that, just as *a* will lead to picture element *p*, so may *b* or *c* or *d* also give rise to *p*. As such, it is possible to manipulate the file such that *visually* the image appears uncorrupted, but *this is not the case* "behind the scenes". Insight into this process will be given later in Section 3.2.2.

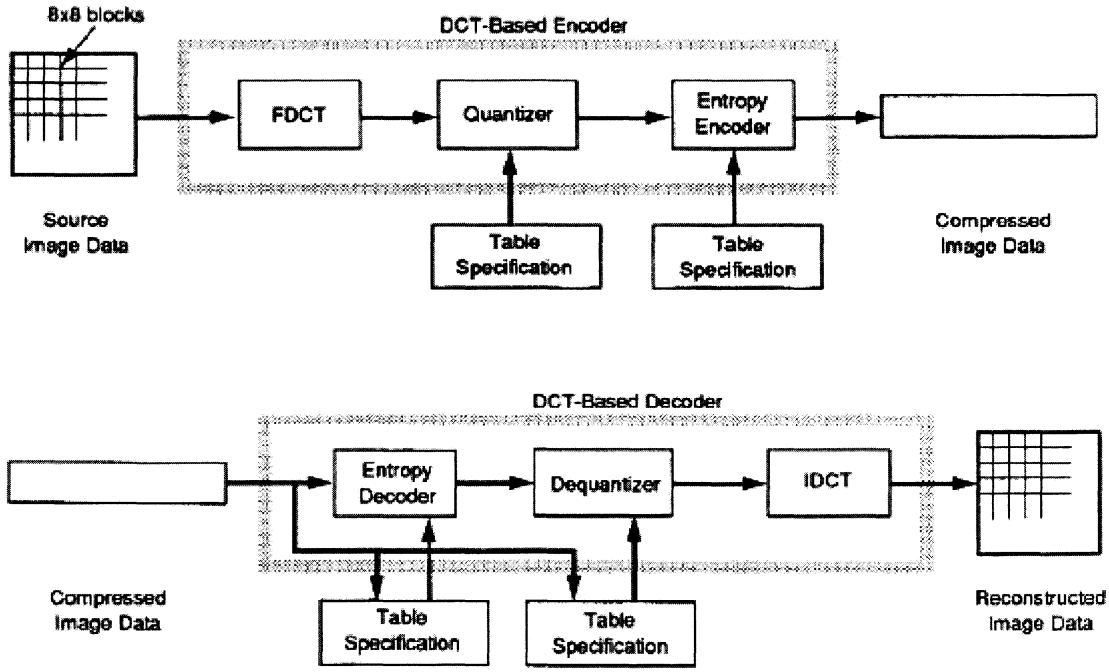


Figure 2.1 - DCT-Based Encoder (top) and Decoder (bottom) (Wallace, 1991)

We have thus far seen how the JPEG file specification accomplishes *en*coding, but what happens when a viewer wishes to *de*code the image data for presentation? Simply stated, decoding is just the reverse of encoding – but the realization must be made that there *is* room for error (or, in our case, data injection) in the DCT table *without image degradation or corruption*. Exactly how will be explained later, in Section 3.2.2.

From the discussion above, it becomes an important point to realize that JPEG files have in fact *two* representations of data for each element – as with transforms of DCT's ilk[12] - the "spatial domain" and the "frequency domain". As such, JPEG images have *two* domains in which data may be covertly embedded – a fact that has not gone unnoticed by either steganography or steganalysis researchers.

2.2.3.3. 4- and 3-LSB Static Encoding in JPEG Images

Kurak and McHugh's seminal work (Kurak & McHugh, 1992) on Least-Significant Bit steganography focuses on hiding one image within a cover image through use of 4-LSB encoding. This means that for every 8-bit pixel in their 256x256 pixel image dataset, the least significant 4 bits (rightmost) of every pixel in the cover image is replace with information from the image that is being hidden. While the approach may seem immediately detectable, the reality of the situation was that, given the images utilized were *8-bit (256 levels) gray-scale* in format, it is difficult for both display device and the human eye to differentiate between one level of gray and the next. Indeed, "about 100 levels is all that [the human eye] can distinguish under ideal circumstances" (Kurak & McHugh, 1992). The authors further go on to explain that, given rough surfaces presented in the image, 4-LSB insertion methods may escape

[12] Rightly so, as the Discrete Cosine Transform (DCT) is actually a cut-down version (only the real component) of the Fast Fourier Transform (FFT) (Marshall, 2001).

detection, but that smooth (or "flat", as they deem them) surfaces may survive visual detection even if the 3 least significant bits are being used to carry a hidden data stream (3-LSB insertion). The methodology utilized here was a serial 4-LSB embedding of each pixel in the secret image into the cover image, with no regard for visual degradation or randomization. If, for instance, one were to become aware of the presence of an embedded secret message, then it would be quite possible to write a program that would capture the last 4 bits of each pixel, and thusly reconstruct the original secret image. Therein, though, lies the rub: Kurak and McHugh set the upcoming stage by stating that "[d]etection of hidden images is expected to be a hard problem" (Kurak & McHugh, 1992).

In response to this, *Attacks on Copyright Marking Systems* (Petitcolas et al., 1998) dismisses this kind of bit-plane replacement signal as "easy to detect". However, it must be kept in mind that the key item that distinguishes Petitcolas et al's paper from the kind of *pure* steganography that this thesis is examining is the realization that, in dealing with a copyright (e.g. watermark) embedding system, *an original image exists in the world* – in this case, digitally. Hence, it is *indeed* trivial to take a difference image histogram, which "overlays" the two images and makes a pixel-by-pixel comparison (by subtracting $\text{Pixel}_{\text{image1}}$ from $\text{Pixel}_{\text{image2}}$) between the two, and come to one of four overall conclusions residing within two cases. It may well be that the difference image is comprised wholly of zero values, meaning that the two images completely match at every pixel. This would result in a completely black image, meaning that either (1) both images are genuine *or* (2) both images are *exact copies* of a falsified image (or, for our purposes here, both are stego-images acting as a cover for a covert message). It also may come about that the difference image is non-zero at some or every pixel. This would yield a difference image with "noise", showing – in the absence of two *different*

counterfeit images – that either (3) the first image is either counterfeit or steganographically injected with a message, or (4) the second image is counterfeit or likewise acting as a carrier of covert information. In fact, even in the absence of a true difference image existing in the world, it is entirely possible to *fabricate* a difference image, as has been put forth in the work of Zhang and Ping (Zhang & Ping, 2003).

Given this fact, it becomes clear that in steganography, a common-cover carrier image must *never* be employed (we will see below, in coverage of Lou & Liu's work (Lou & Liu, 2002), that great pains may be taken to defeat the common-cover carrier attack) – this has the same effect as re-using a one-time pad. Simply put, if an attacker is given enough views (here, only two identical cover images are necessary), it becomes possible to guess at the length of the messages or even attack the encoding or encipherment itself[13]. This being the case, any reasonable approach to steganography will make use of *entirely different* cover images (which will become important to keep in mind later, in Section 3, and if the same cover image *must* be used, then the sender would likely re-digitize the image to take advantage of noise in the hardware (as opposed to the artificial noise that is introduced by Lou & Liu), and would then be free to choose an appropriate unique cover image to use for communication. Aura dubs this process the *selection method of invisibility* (Tuomas Aura, 1996).

2.2.3.4. Dynamic Encoding in LSB Insertion Systems

Static LSB insertion methods are – at best – easily detected. Worse still, unfettered and unchecked static LSB insertions may lead to image artifacts that are visible to the human eye, thus raising suspicion. To counter this, the next step in LSB embedding saw *dynamic* LSB bit

[13] This second notion would give rise to finding a *collision* within the encryption routine. More information on this kind of attack against an encryption methodology may be found in *Computer Security: Art and Science* (Bishop, 2003).

encoding. This has been referred to as "hiding in the shadows", as this method makes an intelligent choice about what embedding bit size to use based upon the *context* of the pixel and its neighbors; this typically results in a good deal of covert data free to lurk in shadowed and contoured image features. It is important, at this juncture, to make note of certain high-level similarities between this and data (image) compression. In both arenas, the systems involved have become increasingly more robust and have worked *in tandem* with the human visual system, rather than against it, to achieve maximum data embedding (or compression) rates at a minimal degradation in the signal itself. For instance, today's steganography techniques (taking lessons learned from compression) are adaptive in their insertion mechanism (Lie & Chang, 1999), and even may make use of shadows and features such as natural contours or curves within the image (Lee & Chen, 2000), thus escaping visual inspection through compliance with the *oblique effect* in human vision ((Gray, Cosman, & Oehler, 1993)).

Lie & Chang's work (Lie & Chang, 1999) took LSB encoding to new levels by *adapting* embedding rates with the image qualities, thus taking advantage of the human visual system. This is a crucial development to make, as it strives to work in harmony with a system that is more keenly able to pick out image artifacts than a machine-based method, depending upon the image context and the level of LSB embedding in use. However, there still existed the problem of providing good capacity within stego-imagery while also maintaining good image fidelity. As such, Lee & Chen (Lee & Chen, 2000) designed a model that achieves just this end. Lee & Chen used a dynamic-length LSB embedding methodology, and achieved greater than 50% capacity (above and beyond 4-LSB encoding), while maintaining image fidelity at a superior quality than a static 4-LSB embedding methodology. This is good news for LSB-based insertion, as it marries together the two seemingly disparate goals discussed

above, being preservation of image fidelity while increasing payload capacity. Yet, for all the good that their method did, it was found to be detectable when looking at the noise generated through compression artifacts. And this is where Lou & Liu come into play. Lou & Liu (Lou & Liu, 2002) further improved upon the Lee-Chen model of dynamic LSB embedding with the addition of Gaussian noise to the untouched image pixels, thus increasing the stego-image's survivability to a common-cover carrier attack. Even though this thesis is not going to use the common-cover carrier attack, favoring instead a more realistic (and much more difficult) zero-knowledge approach, the import of the Lou-Liu extension to the Lee-Chen model of steganographic embedding is not lost here: this effectively takes away a *trivially simple and accurate* tool from an individual conducting steganalysis against the image.

It must be noted at this point that, while LSB encoding schemes are easy to implement and *potentially* resistant to inspection, they suffer from being unsuitable for transmitting large amounts of information. To this end, Lin & Lee's *Confused Document Encrypting Scheme* (Lin & Lee, 1998) could be employed for text files, or Yeh & Hwang's approach (Yeh & Hwang, 2001) may be taken to extend the Confused Document Encrypting Scheme to utilize *any* digital file, while allowing the use of two-byte character-based languages (e.g. Chinese, Japanese, Korean). This research stream is mentioned here for completeness only, and to illustrate that n-LSB embedding of information has definite drawbacks in not only the *domain* of insertion, but also of the ontology itself.

2.2.3.5. Beyond LSB: Frequency-Domain Encoding

Given that even the most robust and dynamic LSB insertion scheme is prone to detection, recent publications have shied away from this topic, instead favoring a more covert mechanism for information hiding. It is important to take note that this relatively new

information hiding routine present in JPEG files is found within the file specification itself: use of the frequency-domain. As such, it is many times more infeasible and onerous to uncover information covertly hidden using this mechanism than using LSB insertions.

Even though frequency-domain encoding is a more insidious method of covert data insertion into (JPEG) images, even this advanced breed of information hiding is not without attacks. In their 2001 work (Fridrich, Goljan, & Du, 2001), Fridrich et al. propose a method of steganalysis that can uncover tampering in the frequency domain – and even provide the potential for bit-level granularity – as well as discover if an image was ever presented in a JPEG format. This method of steganalysis is called JPEG compatibility, as it works with the JPEG algorithm to flag blocks that could not have been created within the confines of the quantizing mechanism, and are likely the result of frequency-domain information embedding. Indeed, for longer messages, it is possible to analyze the image and identify *individual pixels* that have been modified (Fridrich et al., 2001). This is a powerful, heady analysis, and as such suffers from computational intractability; the authors note this in the paper and acknowledge this as a limitation. Even with the computational difficulty of this method, the sky seemed the limit for steganalysis – as most digital images are, or had been at one time, JPEG files, the ability to detect embedding information *to the pixel* is powerful. However, this new-found steganalysis power was not to last long – ending abruptly at the hands of Richard Newmann and his co-authors (R. E. Newman, Moskowitz, Chang, & Brahmadesam, 2002).

Newmann et al.'s paper (R. E. Newman et al., 2002) presents a novel approach to encoding – "[through encoding of] the embedded data in the spatial domain (bitmap) by manipulating the image in the frequency domain (the JPEG coefficients)" (R. E. Newman et al., 2002). The resulting image, as it hinges upon *spatial* information, may be saved as either a

JPEG or a BMP file without loss of (covert) information. This work is similar to that of Marvel et al., who proposed storing information in the quantized JPEG coefficients at a capacity of one bit per block (Marvel, Hartwig, & Boncelet, 2000), but for the fact that this work presents a notion of topological "closeness" as a determinate for data embedding. While this work minimizes detection, so too does it diminish the *payload* that may be embedded at one bit per usable[14] 8x8 block, a 640x480-pixel JPEG file could carry (assuming 95% usable blocks[15]) 4560 bits, or 570 bytes. However, if the payload can be kept small, then involved parties may be assured that this method is a much more robust and transparent one than any previously discussed. Furthermore, this method has the added advantage of not needing any additional domains than the spatial domain – which means that the cover image may be expressed as a BMP file without loss (or undue disclosure) of any information.

Westfeld's F5 algorithm (Westfeld, 2001) embeds bits in the DCT coefficients, thus minimizing the impact to the spatial domain of the cover image. The F5 algorithm marks an improvement over earlier iterations, called – appropriately enough – F4 and F3. F4 came about to replace the idiosyncrasies of the F3 algorithm (embedding of steganographic zeroes in a statistically detectable fashion, and the lopsided odd-even coefficient distribution within histograms of F3-embedded stego-imagery), by mapping zeroes and ones in a less predictable fashion (Westfeld, 2001). The F5 algorithm further improved upon the F4 algorithm by "spreading out" the covert message in the cover image through both permutive embedding (Westfeld, 2001) and through use of matrix encoding (Crandall, 1998) – F5 is regarded as being an early-adopter in this arena. However, F5 – by its very nature – alters the histogram of

[14] The authors did, however, find very few "poor" blocks, so at least the odds favor any given 8x8 pixel block being *able* to support one bit of data.
[15] An arbitrary figure, congruent with "very few" blocks being "poor", e.g. non-useable.

the DCT coefficients, a fact which is potentially exploitable by second-order statistical inspection. Yet the kind of analysis required to accurately and dynamically perform this kind of inspection warrants its own section, below, as this is clearly an *advanced technique* of steganalysis and thus separate from methods discussed to this point.

2.2.3.6. Advanced Techniques of Steganalysis

Currently, with computational power becoming increasingly more powerful and parallelizable, steganography and steganalysis are undergoing a sort of renaissance – this has culminated in the use of steganographic embedding techniques which are easily able to thwart the human visual system as well as first-order statistical modeling. Machine learning and data mining techniques are relatively new public entrants to the art of steganalysis. Rather than subjecting image files to a visual inspection, and then looking at histograms and DCT coefficient tables manually, today's advanced techniques utilize higher-order statistics and *also* making use of the concept of *generalization*, which can tell if image *b* is a candidate for steganographic embedding after being trained on images where the steganographic embedding technique present within *b* was not utilized. In essence, both data mining and machine learning techniques focus principally (for the purposes of image processing) on *feature extraction*, a concept under active research in mathematics.

The first large machine learning initiative came from Lyu & Farid in 2002 (Lyu & Farid, 2002), and makes use of Support Vector Machines. However, the method outlined suffers greatly from problems typical and endemic to automating steganalysis – it relies *heavily* upon pre-processing techniques. In fact, this method has been specifically faulted for focusing on the "complex second order model built of the image" (Berg, Davidson, Duan, & Paul, 2003). This is certainly not to say that it represents anything less than a breakthrough for the

field, as it is powerful; rather, the method is computationally cumbersome and time-intensive. For instance, this method is able to detect steganographic methods that work to correct distortions in first-order statistical distributions or transform coefficients – this fact alone makes the method robust against many current steganographic approaches. As was mentioned previously, this article was the first to make use of a true machine learning technique; one of the authors (Farid) *did*, however, make use of a Fisher linear discriminant (FLD) methodology in an earlier work (Farid, 2002), a method that suffers from its inability to solve non-linear problems. The support vector machine, though, can solve these problems and in doing so may also attain a finer resolution as well as solve problems that the FLD cannot (for a concrete example involving image classification, refer to the improvement of Table 2 over Table 1, in (Lyu & Farid, 2002)).

The prior method discussed, while enjoying benefits of powerful analysis and low false-positive rate (1.0%), will naturally be eclipsed by methods requiring less pre-processing and raw computation – one such method is presented in the 2003 paper by Berg et al. (Berg et al., 2003), in which the back-propagation Artificial Neural Network approach (Rumelhart, Hinton, & Williams, 1986) routinely outperformed (or, in the worst case given, performed comparably to) the data mining methods of decision trees, naïve Bayes classifiers, and a special-purpose program, *StegDetect* (Provos, 2004). Remember from the earlier discussion on Artificial Neural Networks that *overtraining* becomes a real problem in this kind of machine learning technique: an *overtrained* Neural Network is essentially one in which generalization is no longer possible. Berg et al. *certainly* get around this limitation by only sampling a total of 150 images – 50 each of "flower", "mountain", and "tree" types. This is unfortunate, as with this small sample set and similar context, it is difficult to determine to what extent the Artificial

Neural Network is really *working*. Additionally, no information was given as to the naming convention (for instance) of the image files, nor was any information given to the scrubbing of "meta-information" present within the file, such as EXIF data. This is a potentially *massive* oversight, as the ANN learning technique can pick up on nuances like this (if presented), and their black-box nature makes it impossible to determine from *what* inputs, and *how*, its output comes into being. This key limitation aside, it is important to view this article in its more lax context: current steganalysis tools are becoming more complex, more adaptable, and better at detection of steganography.

The solution presented by Berg et al. (Berg et al., 2003) is just one example of neural network technology being brought to bear on the problem of steganalysis: another such example also comes in 2003 (Shaohui, Hongxun, & Wen, 2003). A back-propagation Artificial Neural Network is utilized here, just as in Berg et al.'s article; it is so, quite frankly, because *it works well* for the task of feature extraction and image classification based on patterns and trends (it is for this reason that ANNs find use in all sorts of pattern-recognition tasks from weld inspection to leather and fabric process troubleshooting). Each image possessed 40 discrete statistics, and the ANN was trained on Chen's quantization index modulation method (Chen & Wornell, 2001) of information embedding. However, as with the previous use of Artificial Neural Networks, this article also suffers from a lack of appropriate sample size (training on 23 and 21 images for "clean" and "stego-imagery" respectively) as well as a perhaps inappropriately small test dataset of only 40 (clean) and 41 (stego-) image files. The model accuracy displays precisely this criticism, achieving a false negative rate of 14.6%, and a false positive rate of 25% (see Table 1 in (Shaohui et al., 2003)). The following year, the authors again published regarding the use of neural networks for image steganalysis

(Shaohui, Hongxun, & Wen, 2004). This time, however, the method was more keenly focused on wavelet texture analysis, utilizing the wavelet image transform much like Lyu & Farid's 2002 work (Lyu & Farid, 2002), discussed above. The DWT (Discrete Wavelet Transform) method is beyond the scope of this thesis, but suffice it to say that wavelets provide a contextually-rich analysis of the image, with use in image processing of (at least) analyzing "smoothness" of a given target.

While current technologies in steganalysis tend to focus upon machine learning approaches, it is important to bear in mind that there *still* exists utility for alternate steganalytic methods. For instance, image quality metrics (Avcibas, Memon, & Sankur, 2003), while currently out of favor, may certainly still be used to varying degrees of success. This is an important consideration to bear in mind, as it can be helpful in preventing research stream lock-in and subsequent myopia with regard to other techniques and "best practices".

3. RESEARCH METHODOLOGY

3.1. Introduction

Digital image steganalysis is a particularly difficult endeavor, as it crosses the major research domains of cryptography, image processing, and applied mathematics. Many models – both seminal and current – tend to focus upon *one* particular type of (typically academic) steganography to attack, and go to great lengths to create a large and cumbersome routine that will defeat this steganographic technique. While this approach makes for the academic advancement of the field as well as (rightfully so) bragging rights, these approaches typically do not provide much aid to the individual or corporation who may desire to uncover steganography present within image files in their possession. These approaches are – as measured both in CPU-time and deployment time – cumbersome and unwieldy, and the current state-of-the-art approach also relies heavily upon extensive and lengthy first- and second-order model-building to make a *full* and *contextually rich* description of the image file to the system (for a more thorough discussion of the faults in such a system, please refer to Berg et. al. (Berg et al., 2003), or back to Section 2.2.3.6).

While these types of models have their place, this thesis aims to take a *minimalist* approach that is not found in the body of literature surveyed. This thesis, moreover, presents the following composite research question: "*how little* data is necessary for a data mining approach to give results that are both *reasonable* and *fast*, and *how flexible and extensible* is this model with respect to variable steganographic embedding techniques?". This is a valid and important concern, as well as a potentially viable path to take, given the fuller potential

context of the PIRANHA classification system as working with pluggable, small, and specialized blocks of analytical code rather than one monolithic classification routine.

This thesis takes a relatively difficult road, as it focuses upon a *zero-knowledge* view of the image files. This means that there is *no* use of difference images, and also that, outside of *training* the artificial neural network, there exist no processes subsequent to that which "steg-impregnates" image files that actually *know* if the image is a cover or stego-image or if the image is instead benign. In fact, given the quality and properties of the vast majority of the files chosen (see Section 3.5.3), a human observer will also be hard-pressed to detect many of the stego-images as having anything untoward within them.

Finally, it must be noted that the models generated and presented within this thesis have been created with the end-goal of classification run-time in mind. As such, the models that follow will *intentionally* err in favor of speed over accuracy, and will aim to present – with very broad and quick strokes – information that may help an observer determine *when* it may be potentially viable and worthwhile to utilize one of the more specialized and cumbersome steganalytical techniques discussed in Section 2.2.3.6.

3.2. Image File Description

3.2.1. JPEG File Specification

As has been previously mentioned in Section 2.2.3.2, the JPEG file specification enjoys wide use for electronic imagery, due principally to its lossy compression routine providing typically high image fidelity and small file sizes. Of course, it is through the use of a *quantizer* (refer to Figure 2.1 and also (Wallace, 1991)) that JPEG imagery serves the needs of

steganography by providing a one-to-many mapping of values. Additionally, though use of the Discrete Cosine Transform (DCT), the image is actually comprised of *two* representations – that of the human-perceivable bitplane in the spatial domain, and its analogous DCT-perceivable coefficients in the frequency domain. As was ominously noted at the end of Section 2.2.3.2, this fact "has not gone unnoticed by either steganography or steganalysis researchers" – this thesis makes *exclusive* use of JPEG imagery due to both their wide acceptance and use as well as their additional properties that make them an ideal target for use as a covert communication channel. Furthermore, the use of a compressed image representation such as JPEG is fully compatible with Eggers, Bauml, and Girod's assertion that "uncompressed image data looks to Eve (*the observer*) as suspicious as encrypted data. Thus, the steganographic image r has to be always in a compressed format." (Eggers, Bauml, & Girod, 2002).

3.2.2. Image Domains

To this point, the notion that JPEG images have two domains has been discussed, but not explored. To wit, it is helpful to envision a block diagram, pictured in Figure 3.1 and Figure 3.2, when thinking about JPEG imagery. We will begin by looking an overview of the DCT process in Figure 3.1, and follow this with an overview of the IDCT process in Figure 3.2.

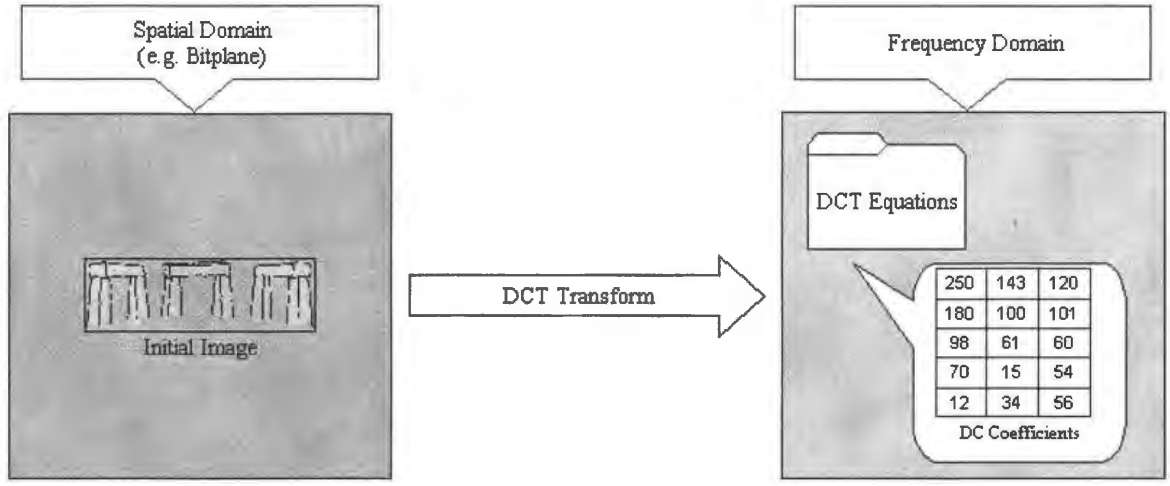


Figure 3.1 - DCT Transform Process: From Bitplane to DC Coefficients

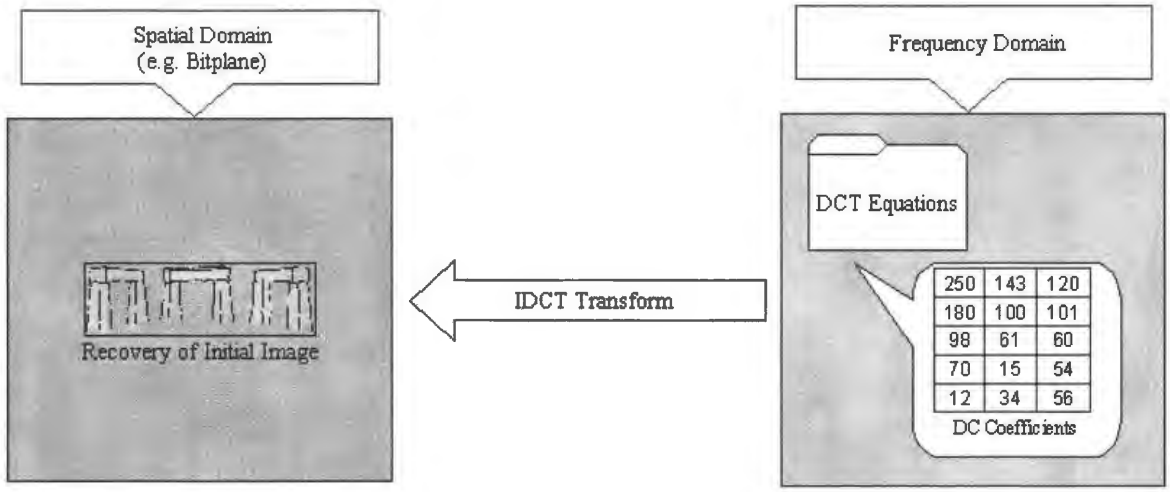


Figure 3.2 - IDCT Transform Process: From DC Coefficients to Restored Bitplane

As may be seen at a high level, the physical representation of the bitplane (e.g. in the Spatial Domain) is encoded into a series of equations in the Frequency Domain when a file is moved into JPEG format. Correspondingly, when the file is called upon for display, the

process is reversed by virtue of the Inverse DCT function, and the initial image is restored in the bitplane[16].

3.3. Transforms and Signal Processing Used

Steganography, as has been touched on above, occurs when a signal makes use of a cover or "carrier" object to mask its presence. In this thesis, as JPEG files have been selected as the cover object, there exists the potential for image processing and transformation techniques to be used.

3.3.1. Discrete Cosine Transform (DCT)

The Discrete Cosine Transform (DCT) is an integral component of the JPEG file specification. As noted before, it is a variant of the Fast Fourier Transform (FFT) that only includes the real components. Before launching into the equations that govern this process, it is handy to envision a block diagram of the Discrete Cosine Transform (DCT) process, as well as the Inverse Discrete Cosine Transform (IDCT) process that recovers the spatial-domain bitplane. In the following figures (Figure 3.3, Figure 3.4, Figure 3.5), let f denote a bound matrix of size $M*N$.

Figure 3.3 - g as a DCT of f (Dougherty & Giardina, 1987)

[16] A word here is required about surreptitious manipulation of the DC coefficients. Then, while the image may well *look* the same and *perform* the same in the bitplane (by virtue of the quantizer), it will not be *fundamentally* the same image. This is a small point, but one that must be made salient.

Figure 3.4 - Recovery of *h* after IDCT(*g*) is executed (Dougherty & Giardina, 1987)

Figure 3.5 - When *f* is saturated, *h*=IDCT(DCT(*f*))=*f* (Dougherty & Giardina, 1987)

With this broad overview in place, it is now possible to examine the transform at a lower level. The DCT of *f* is formally given here in Equation 3.1:

$$g = AfA'$$

Equation 3.1 – *g* as a mathematical expression of DCT(*f*) (Dougherty & Giardina, 1987)

A', above, is the transpose matrix of *A*. Matrix *A* has entries as follows below from Equation 3.2:

$$a_{jk} = \begin{cases} \dfrac{1}{\sqrt{n}}, for \quad all \quad k = 0 \\ \sqrt{\dfrac{2}{n}} \cos\left[\dfrac{(2j+1)k\pi}{2n}\right], for \quad \begin{matrix} k = 1,2,\ldots,n-1 \\ j = 0,1,\ldots,n-1 \end{matrix} \end{cases}$$

Equation 3.2 – Definition of Matrix *A* (Dougherty & Giardina, 1987)

With the above notions contributing to a more complete understanding of the process, the DCT function may more simply be expressed as g = DCT(f) (Dougherty & Giardina, 1987). The equation that dictates a two-dimension DCT (as would be found in imagery) follows in Equation 3.3:

$$H(u,v) = \frac{2}{\sqrt{MN}} C(u)C(v) \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} h(x,y) \cos\left[\frac{(2x+1)u\pi}{2M}\right] \cos\left[\frac{(2y+1)v\pi}{2N}\right]$$

Equation 3.3 - 2D Discrete Cosine Transform (Crane, 1997)

Very briefly, it may be said that images may be broken down into a set of basic functions with the DCT – this implies, therefore, that an image's bitplane presentation in the spatial domain is embodied by a summation of basis functions present within the frequency domain. I would like to talk very briefly about the meaning of the word "basis" used in the above sentence; its inclusion is *not* a typographical error. Rather, basis functions come about from the realization that if values can be *pre-computed* for given *N* and *M* values, then "blocks" of input signals may be transformed much more rapidly than finding these end values through computation. As JPEG is a block-based image routine, this principle holds here. JPEG makes use of 8x8 pixel blocks, which are then fed into the convolution mask that results from pre-computation – resulting in a vastly improved runtime for the transform. The DCT basis functions are illustrated in Figure 3.6. It is important to further realize that these pre-computations essentially represent weighted averages of the DCT functions. Compression may be achieved by weighting more of these averages as "0", or image fidelity (at the expense of size) may be realized by setting these weights away from 0 and closer to the actual value of their constituent pixels. With this realization made, it becomes a trivial cognitive task to see that where compression may be achieve, so too may data hiding. In fact, working within the JPEG file specification in this manner, Derek Upham's *Jsteg* algorithm (Upham, 1999) achieves covert data hiding.

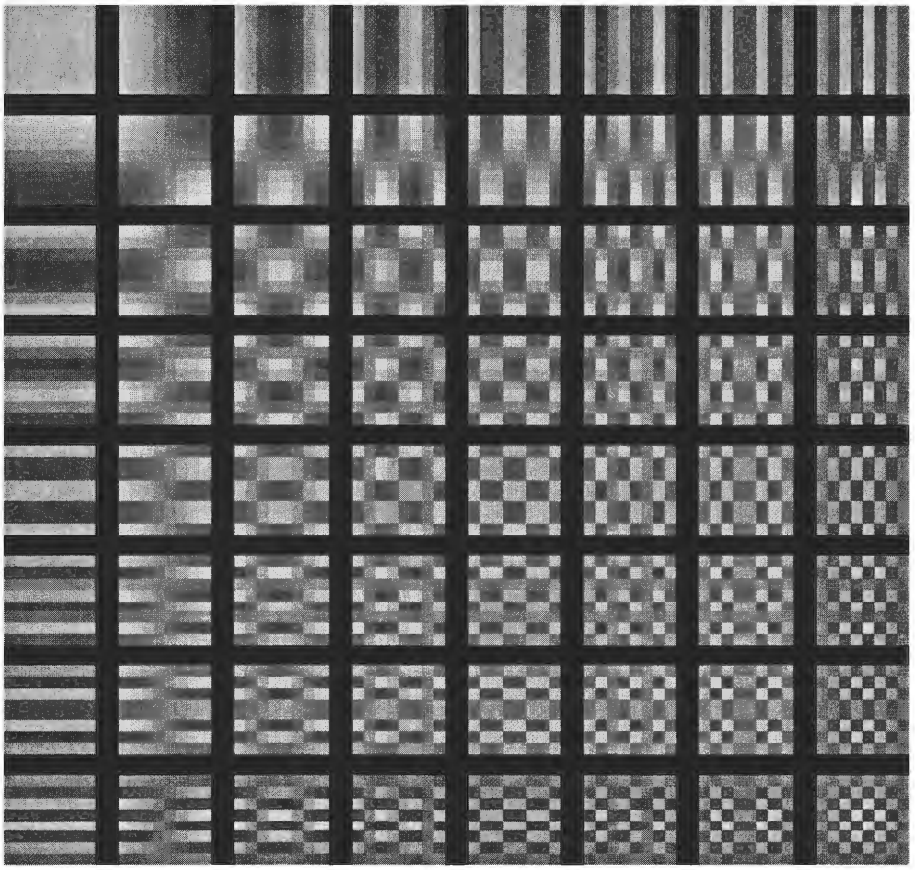


Figure 3.6 - The 64 DCT Basis Functions for an 8x8 Matrix

3.3.2. Histogram (grayscale values)

An image histogram is a plot of the color value (here, in an 8-bit grayscale image file, this is within the bounds of [0,255]) against the number of pixels having that particular color value. For instance, a black-and-white (1 bit) image would yield a histogram that is essentially a bar graph with only two categories – the "height" of the bars would represent the numbers of pixels having a particular color value (white or black in this example), and the individual bar would correspond to the color value itself. Here, however, the data are diverse and varied enough in both color value and count to allow for meaningful statistical measures to be conducted on it. For instance, the simple measures used here for the histograms of images

were minimum, maximum, variance, and standard deviation. These values create a rough approximation of the distribution of color (gray values) within the image, and are useful even here. I say *even* here because there are *specific* and *known* steganography algorithms (one example that comes immediately to mind is *jsteg*, (Upham, 1999)) that rather radically alter the histograms of the images that they use as cover-images.

3.4. Equipment

3.4.1. Personal Computer Used

For the research, a stand-alone personal computer was used. The hardware consisted of a Pentium® 4 3.0GHz processor running on an ASUS motherboard with 1GB of RAM and 250GB hard disk space. With the problem space being potentially vast – with respect to sample size, number of image attributes, and epochs through training – this level of COTS hardware, while strictly more than is absolutely necessary, helps ensure a "reasonable" runtime with each iteration through training.

3.4.2. Conversion, Analysis, and Modeling Software Used

Image files were converted to 8-bit grayscale 75% quality JPEG files – both with and without optimal Huffman encoding – through use of Advanced Batch Converter (Gold-Software, 2004). 8-bit grayscale images were used as a result of current "best practices" that have evolved out of a need to minimize the impact on the human visual system. 8-bit grayscale imagery meets this demand "because grayscale values change gradually from palette entry to palette entry" (T. Aura, 1995), whereas color images typically do not see this 1- or 2-shade palette differentiation. In instances in which image rotation was necessary, Advanced Batch

Converter was also utilized in a two-pass manner consisting of rotating images and then, after copying these rotated and resized images to another directory, resizing the remaining images and merging the two image sets. While this is a cumbersome constraint to endure, the software did not enable the two to take place in the same pass. Although this is an important (and time-consuming) drawback to be aware of, the software performed the task of "batch converting" the image files admirably. Matlab Student Version 14 with SP1 (The MathWorks, 2004) was used to read in the image and create image attribute matrices, as well as harvest DCT coefficients from the images. In addition to the Matlab core component, the Image Processing plugin was used, to provide for some finer-grained analysis as well as DCT extraction. Matlab is a well-known and trusted tool for engineering problems, and is routinely used for image analysis and model-building, which makes it an excellent candidate for this kind of work – while a tool to perform image analysis in this thesis *could* have been written from scratch, such "home-grown" tools have not been iteratively scrutinized for years as Matlab has. Instead, however, of using Matlab to build the predictive model, Weka (Witten, Frank, & Kaufmann, 2000) version 3.4.3 was used to both grow and validate the Artificial Neural Network model that was created in this thesis. Weka has a fairly well-defined track record as being a serviceable and worthwhile tool for data mining, and has the added advantage of being provided free of charge.

3.5. Image Preparation

3.5.1. Image Description

Initially, a set of 747 images was gathered from known "good" sources. "Goodness" here presents the notion that these images were guaranteed to be free of prior steganographic message embedding. This is an important constraint to ensure, as having an image set that is *known* to be free from steganography is vital to the validity of the models built, below. Gathering nearly 1,000 of these images was no easy task, and I thank the individuals listed below who kindly gave me use of their images in this project. Images consisted of the following, listed in descending number of images provided:

- JPEG images of England and Reiman Gardens, taken by myself over the summer of 2004.
- JPEG images of varying architectural and natural scenes, expertly photographed by Phil Greenspun[17].
- TIFF images from the USC-SIPI Image Database (California), a *de facto* standard purveyor of digital "test" images.
- JPEG images, provided courtesy of Juan C. Melendez, taken on vacation in Europe.
- JPEG images, provided courtesy of Tom Hauppenthal and taken by him in the summer of 2003 while on vacation.
- JPEG images of Iowa State University, taken by myself upon arrival in Ames, Iowa.
- JPEG images from Compuworks' **Super Graphics Collection** CD.
- JPEG images from Corel's **Professional Photos Sampler** CD.

[17] Images were used with the express and kind permission of Philip Greenspun, at http://philip.greenspun.com/

3.5.2. Image Size

Images were resized to 640x480 pixels or 320x240 pixels. These file sizes represent a balance between both realism and image manageability, respectively. These sizes are also typical with respect to consumer-grade digital photography. Additionally, having all images a standard size allows for Kurak's 1992 Image Downgrading (Kurak & McHugh, 1992) mechanism to be utilized in the form of Matlab code[18], wherein a cover image masks a "secret" image, using pixel-by-pixel *n*-LSB embedding. Finally, this approach also prevents, in the event that an interested party wanted to train an Artificial Neural Network or SVM to look at the pixel values of the spatial domain themselves, the first few neurons in the Artificial Neural Networks used from becoming *over*trained while the last few neurons would be *under*trained. While many images were rotated (to keep with the 640x480 pixel profile), some images retained their portrait (480x640) layout.

3.5.3. Image Context

Images chosen consisted largely of natural scenery, due both to their textured quality – which, remember from the discussion in both Section 2.2.3.3 (stemming from (Kurak & McHugh, 1992)) – as well as survivability from inspection by the human visual system (from Wayner, below). Whenever possible, faces and skin-tones *were not* utilized, as the human eye is expert at spotting skin-tones and facial features that are unnatural. This truth, in turn, sharply contrasts with the human visual system's *tolerance* for examining leaves in a tree that are one or two shades "off" (Wayner, 1996). The images were stripped of EXIF information (which is meta-information contained within the image file; typically, this can contain information about the camera or program that was utilized to create the image), as inclusion of this information

[18] Adapted from Fabien A.P. Petitcolas' Matlab code, electronically available at http://www.petitcolas.net/fabien/steganography/image_downgrading/code.html.

and its use might bias the Artificial Neural Network in its weighting and decision-making metrics and therefore skew the results to either favor or abhor images with EXIF information present.

3.5.4. Image Selection

Images were hand-selected by the author to present both a wide variety of image "texture" (see Section 2.2.3.3) as well as a sound representation of images that may be found in current circulation and used as innocuous carriers. Fractal and computer-generated images were *not* used, as these equation-constrained and computer-generated images have statistical properties that may be rather predictable, and may end up skewing the output. Images with large, smooth areas of either sky or water were, on the whole, avoided. All individuals present within the images used in this thesis gave their express consent to the author, and I again wish to thank them for their time and likeness. It is important to take note that *none* of the images were "histogram-equalized", as this would make analysis largely invalid through the creation of a data mining rule that would predict "stego-image" whenever a non-normal histogram (derived from the Min, Max, Mean, and Standard Deviation of the histogram) was presented to the data mining system. This does, however, make for some images which are a bit more dark than they "ought" to be, or otherwise similarly uneven in their contrast. This is intentional, as it both preserves data quality and upholds the results obtained from data mining, *and* also is representative of many consumer-grade digital imaging devices and users – indeed, the author himself does not attest to be much good with a camera, and suspects that most causal users of the technology likely fall into this category.

3.5.5. Image Warehousing, Format, and Storage

Images were first brought into the system from a variety of sub-directories ("folders") in a specified thesis directory on a computer hard drive. This folder, and all the images within it, were then marked "read-only" to prevent possible tampering or loss. After the images were warehoused on a personal computer, the files were converted to grayscale and resized and then written to a directory that was named according to the size and JPEG options present within the files (e.g. "640x480 greyscale with optimal huffman codes" was a directory that housed images used in the analysis, as was "320x240_greyscale_without_optimal_huffman_codes"). All images were saved with an initial JPEG quality of 75, which is representative of the vast majority of typical JPEG imagery on the Internet and in personal use, and were varied according to size (640x480 or 320x240 pixels) and Huffman code ordering (optimized or not). Additionally, *all* resize operations took place from the original image, and *all* steganography operations were taken on the *derived* image – this helps combat unnecessary image degrading and quantization effects, respectively.

3.5.6. Image Naming Convention (and why this is important)

The images, in their respective directories, were renamed [1-747].jpg. This is an important step to take, as it enables programmatic examination of all image files in a directory through opening the directory, and then looping from 1 to 747 (Matlab 1-indexes loops) to examine each file individually. This makes for a rapid and lightweight program, and does away with complex and cumbersome (and potentially dangerous) string parsing. Additionally, this also aids in the creation of a meaningful and readable output file, as the record number will relate exactly to the image that the record represents (e.g. the 49^{th} element in the dataset will correspond to attributes gathered from the file "49.jpg"). This makes it easy to track and

explain image statistics, and also lowers the bar of entry for an outside observer to make sense of the Matlab code used to generate the summary statistical measures and output files.

3.6. Model-Building and Validation

3.6.1. High-Level Description and Process Flow

For this thesis, two "image steg-impregnating" Matlab modules were created, as well as modules to analyze all files in a directory and then roll these observations into a comma-separated file for later data mining and analysis. At a few points in the Matlab modules, the choice was made to favor *ease of use* over *runtime* by actually invoking a shell to run the steg-impregnating routine (e.g. Hetzl's *steghide* (Hetzl, 2003)). This approach, where taken, supersedes a more primitive batch-file creation, as by creating a shell instance within Matlab, error detection (e.g. generic file-system "file not found", or task-specific "steghide: the cover file is too short to embed the data" errors) and – admittedly – primitive correction is implemented. Where this approach is taken, it is indicative that there exists a non-trivial chance of the embedding routine to fail. Modules are intentionally kept specialized and small, in an effort to increase future viability for the extension of PIRANHA into a classification system that is supported by modules that are specialized, fast, and discrete while at the same time being self-supporting and sufficient. Currently, two small modules have been written to impregnate (dubbed "poison" or "infect" in the comments of the Matlab programs) images – these two modules make use of the image downgrading method outlined in Kurak (Kurak & McHugh, 1992), and Hetzl's *steghide* program (Hetzl, 2003), and are named *Steg_nLSB.m*[19]

[19] For a file listing, please refer to Appendix A.

and *Steghide.m*[20] respectively. The creation of these modules was deemed necessary to *automate and randomize, without prior knowledge or bias with respect to the image file*, the steganographic embedding of a covert signal (an image file in *Steg_nLSB*, and a text file – in fact, 2Kb of text from the Literature Review of this thesis – in *Steghide*).

In addition to handling and tracking individual file infection, the m-files built also handle sundries of directory and file checking and output directory and file creation. This last portion is an important observation to make, as with the naming conventions in use (context-specific and rich with regard to image contents), having the program control the naming scheme greatly reduces human error from inadvertent mis-classification. For instance, an operator may be thinking about the last run and lose track of the test set being encoded with optimized Huffman values or not – this may lead to a gross mis-classification that would result from running a testing set from *A* against an Artificial Neural Network trained on *B*. In short, attempts have been made to remove the human from the loop and automate where possible – yet a keen eye was kept turned toward providing meaningful, intuitive, and engaging output. In this vein, visualization and output from the tool is limited, to err on the side of cognitive simplicity. Examples of this philosophy are demonstrated in Figure 3.7, Figure 3.8, Figure 3.9, and Figure 3.10.

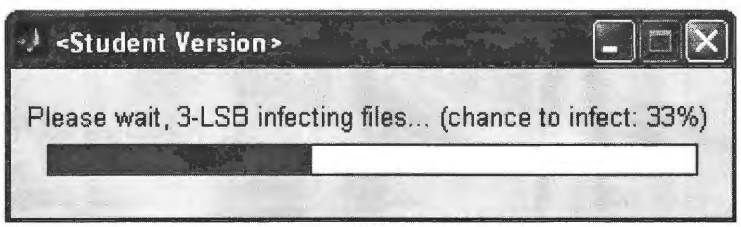


Figure 3.7 – Steg_nLSB.m Process Visualization, 33% Infection Rate (3-LSB insert)

[20] For a file listing, please refer to Appendix B.

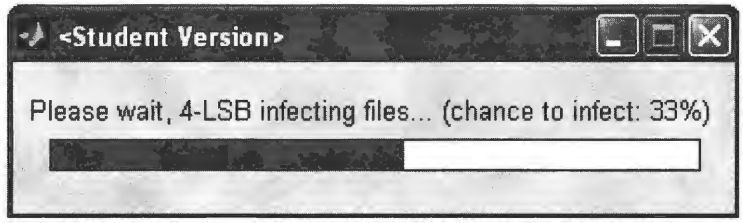


Figure 3.8 - Steg_nLSB.m Process Visualization, 33% Infection Rate (4-LSB insert)

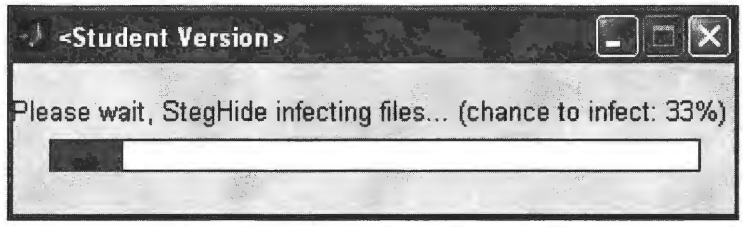


Figure 3.9 - StegHide Process Visualization

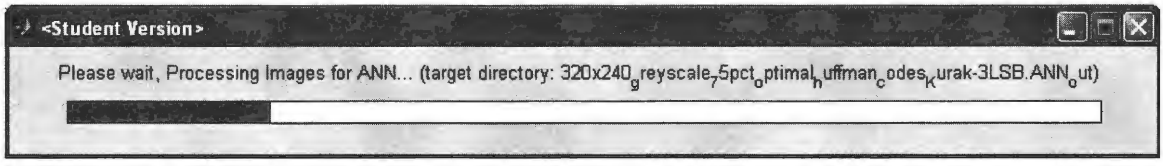


Figure 3.10 - ANNPreProcess.m Process Visualization (note: Matlab subscripts "_x")

Currently, the files that generate the above listed types of cover images have a multitude of user-defined options (such as image directory, how many bits to use, and so on), and *do not* copy the "clean" files from the set in question to the target directory. This is an important observation to note, as it means that, for now, there is still a bit of "hands-on" work involved. However, these programs are fluid and undergoing revision – as such, future versions will contain a multitude of time-saving improvements and run in a more *laissez faire* fashion.

After images are "infected" with a chosen steganographic method, the two methods discussed above write a file, *infected.csv*, to the target directory – this file is a comma-separated file and represents a list (vector) of all the files that have been infected in the form of integer values. A word here is necessary about the randomization routine used to choose images to

infect: simply comparing the absolute value of a random number in the range of (0,1] (the former bound being programmatically enforced) to the user-defined variable CHANCE_TO_INFECT (in the range of [0,1], with 0.33 – 33% – a typical choice) and seeing if the file matching a randomly-selected file (between 1 and 747 here) should be infected *is not sufficient.* It was discovered early on that, without clamping, Matlab would happily – although statistically improbably – infect *every* file in the directory without regard to the bounding constraint (through the use of a logical check of *abs(rand) > CHANCE_TO_INFECT* without further restriction). As such, a more rigorous test involving a comparison between both the "chance to infect" condition *and* the total number of infected files being capped at (*CHANCE_TO_INFECT*747*) of the total files – in short, *enforcing* the number of infected files to remain under a user-defined infection percentage. The Matlab code that achieved this used a bitwise AND that is used by a unary selection *if* statement. It is after this process has completed that the *infected.csv* file is written to the chosen output directory. At this point, the onus is currently on the end user to copy the remaining files (the *uninfected* files) in the current image set (e.g. 320x240 *with* optimized Huffman values; 640x480 *without* optimized Huffman values…) over to the target directory defined in the chosen steg-impregnating routine, choosing to *not* overwrite existing (the *infected*) files in the directory. Finally, the results (image files) must be analyzed and a file suitable for data mining must be generated from the analysis. This process takes place within the m-file module called *ANNPreprocess.m*[21]. This file, as I have broadly defined it doing in an in-file comment, is "Used to preprocess JPEG images for running through an ANN". It is in *this* file that the image processing calls – such that they are – take place, and it is in *this* file that simple and efficient statistical measures of

[21] For a file listing, please refer to Appendix C.

the images are created. Finally, this method creates output that will be used in the next stage of the problem.

The output consists of several files: a comma-separated header file involving names of statistical measures used in the analysis of spatial-domain elements (*bitplane_headers.txt*), to be used as the header of a comma-separated file containing the data from these statistical measures (*{Input Directory Name}.bitplane_stats.csv*), and their corresponding analogues which make use of image histograms (*hist_headers.txt* and *{Input Directory Name}.histogram_stats.csv*). These files may be analyzed separately or, for enhanced detection, in conjunction with each other (this step requiring a manual cut-and-paste from one file to the other). Finally, these resulting comma-separated files are utilized by the data mining program (here, Weka).

In keeping with the minimalist approach, the files are comprised very simply, as follows in Table 3.1:

Generalized File Name	**Number of Observations (Rows)**	**Number of Attributes (Columns)**	**Attributes**[22]
.bitplane_stats.csv	(input images)	7	Entropy, Mean, StDev, VMax, VMin, VMean, StDev2, CLASS
.histogram_stats.csv	(input images)	5	HistMAX, HistMIN, HistMEAN, HistSTDEV, HistVARIANCE, HistCLASS
.merged_stats.csv	(input images)	12	Entropy, Mean, StDev, VMax, VMin, VMean, StDev2, HistMAX, HistMIN,

[22] Not including the *BugFixForWEKA* column that is removed prior to processing.

			HistMEAN, HistSTDEV, HistVARIANCE, CLASS

Table 3.1 - Composition of Data Mining Input Files

It should be noted that these files include a "dummy column" labeled *BugFixForWEKA*. This is intentional, as a bug (unreported, to my knowledge) was found in my experimentation with Weka – the last attribute in a data file *cannot* be converted from a numeric value (such as the "0" and "1" scheme in-use here, which relate to the observation coming from a file that was either known to be innocuous or known to be a stego-image, respectively) to a nominal value – which, as it happens, is precisely the data type necessary for use of an Artificial Neural Network (or any number of classification schemes) in Weka. Hence the inclusion of the "dummy attribute" for Weka – this ensures that the classification (steg-infected versus innocuous) variable moves to the *second-to-last* attribute value in the dataset, and enables it to be converted to a nominal value, which may then be examined by Weka.

One last note must be made about this process: the resulting data mining set is found to yield consistent results *despite* dataset ordering, thus providing support for the premise that the model is indeed providing consistent analysis based upon the selected and reported image attributes. The end-to-end process, involving three distinct stages, is graphically outlined in Figure 3.11:

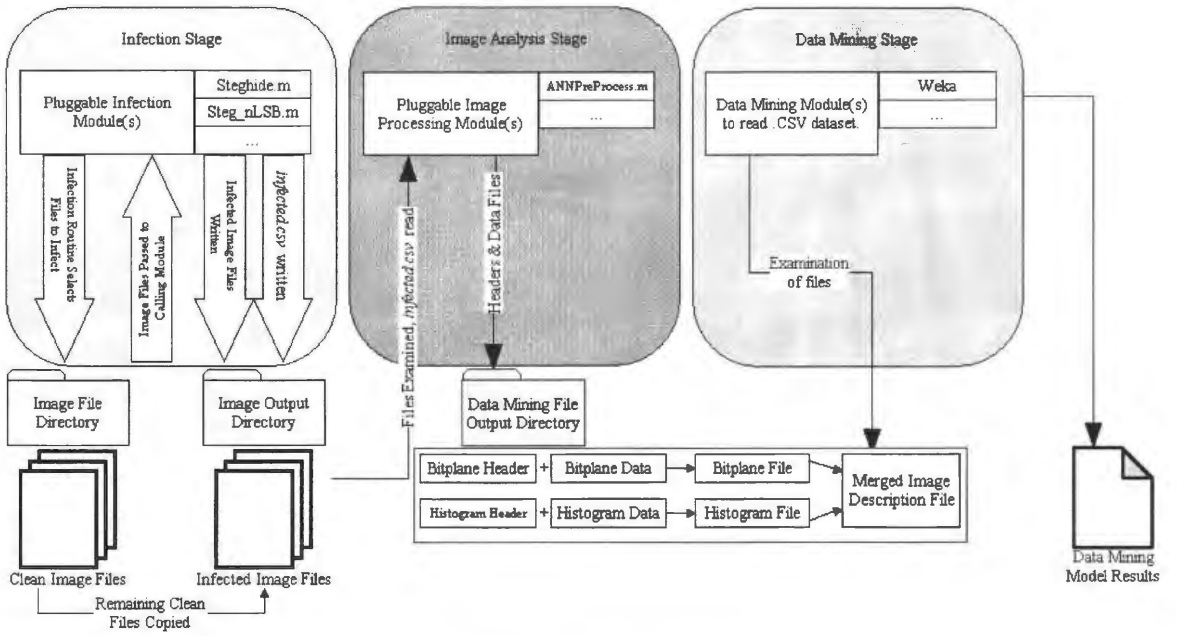


Figure 3.11 - Process Overview

3.6.2. Artificial Neural Networks as a Methodology

Artificial Neural Networks were chosen as a data mining and classification methodology due to their (inherent) flexibility and ability to pick up sometimes obscure patterns, and also their use in classical image processing. Artificial Neural Networks work in a manner roughly analogous to a biological neural network, and a BackPropegation Artificial Neural Network, such as was used in this thesis, is able to "learn" by back-propagating errors through the network and adjusting weights of between-node synapses. The number of nodes – along with the number of epochs, or iterations, that the neural network is trained over – determines the complexity and subsequent accuracy of the network. An ANN with zero hidden layers, remember, is known as a *Perceptron*, and is unable to correctly classify the outputs for linearly inseparable problems, such as the functioning of an XOR logic gate. The number of hidden layers in an Artificial Neural Network, after the first, do not typically add to the overall

accuracy of the network. The total number of hidden layers in the neural network used in this thesis was 1, and each neural network had a total of $\left\lfloor \frac{(Attributes + Classes)}{2} \right\rfloor$ (here, this varied between 6 and 7) nodes, depending upon the number of attributes in the data set. An Artificial Neural Network representative of those used in this thesis (with attribute names) is shown in Figure 3.12.

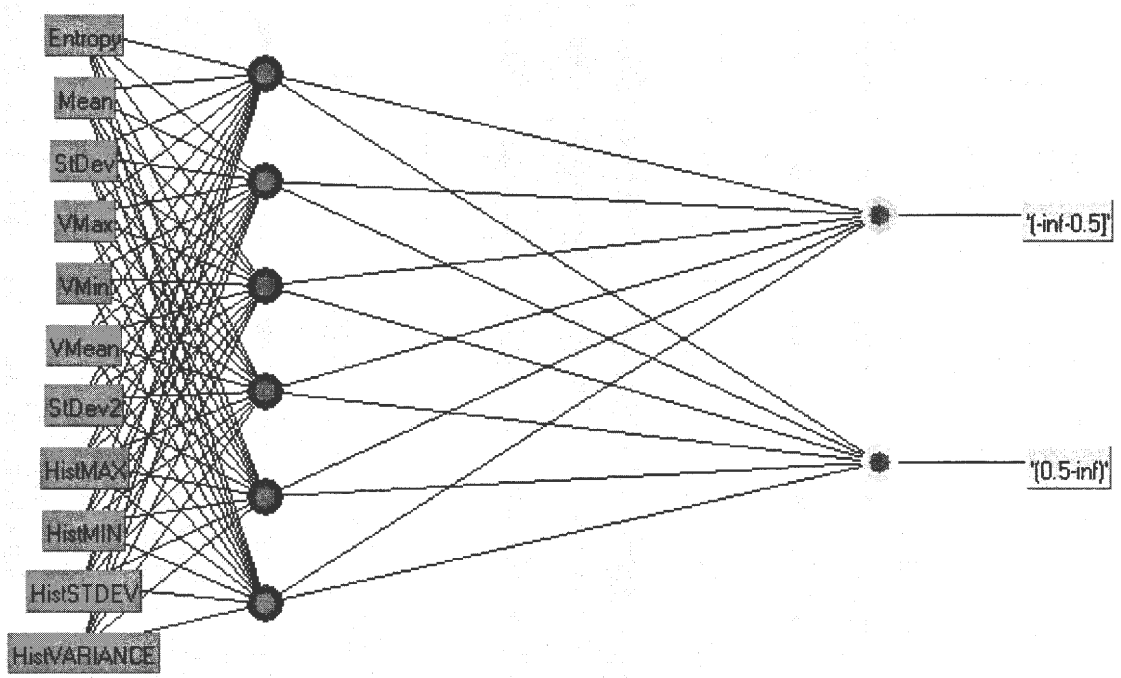


Figure 3.12 - Typical Artificial Neural Network Used

4. RESULTS

4.1. System Output and Analysis

4.1.1. Runtime

Runtime is an important consideration in an application such as this, and as such has been tracked and reported. Using Kurak's Image Downgrading LSB insertion routine, performing 4-LSB insertions into a JPEG file without optimized Huffman encoding takes the most time, and performing 3-LSB insertions into a JPEG file without optimized Huffman encoding takes the least amount of time. This is graphically depicted in

Table 4.1 via the minimum, mean, and maximum runtimes respectively. The problem set was then sent to the ANN Pre-Processor, where image processing measures were recorded to comma-separated files for later analysis. Appendix H demonstrates that the relationship between data type and time taken uncovered above remains unchanged in this operation. The full data, with standard deviation (in seconds; not shown in Appendix F or Appendix G) for the insertions, were as follows in Table 4.1.

Image Embedding via Image Downgrading					
LSB	Optimal Huffman	Min	Mean	Max	StDev
3	1	0	0.045264	0.39	0.065666
4	1	0	0.05043	0.281	0.071126
3	0	0	0.043424	0.266	0.061044
4	0	0	0.058083	0.422	0.083564
ANN Pre-Processor Time					
LSB	Optimal Huffman	Min	Mean	Max	StDev
3	1	0.031	0.047775	0.157	0.011826
4	1	0.031	0.047898	0.109	0.011097
3	0	0.031	0.049111	0.109	0.011438
4	0	0.031	0.052186	0.203	0.015229

Table 4.1 - Min, Mean, Max, Standard Deviation of runtimes for LSB Insertions

These times are given in seconds, and refer to the total amount spent in the working loop of the program.

4.1.2. ANN Modeling Accuracy

There is, of course, much more to the time taken by the system than just runtime. There is also the question of model-building, which is potentially lengthy using an Artificial Neural Network.

The resulting comma-separated data files relating to the bitplane and histogram representations of the image were then manually merged into a data set having columns comprising of all attributes from the bitplane file and all histogram attributes from the histogram file. This was done because it was found that the lift and gain in the model made this worthwhile, and the extra information may be used to derive information that neither set could predict on their own. 5-fold validation was used in the creation of the Artificial Neural Network, and each ANN was trained for a period of 5000 epochs. 5-fold validation ensures model robustness by using 1/5 of the training set as a holdout, and rotating this holdout throughout the training cycle. This ensures that any local oddities have a chance to "work themselves out", and is a commonly-accepted method of both proving and increasing model robustness in data mining. However, nothing comes free – the time involved in training an ANN in this fashion is onerous compared to more traditional methods of data partitioning to create a testing, training, and (optionally) holdout set. A training period of 5000 iterations (epochs) seems a good figure[23], as observation of the ANN has shown that at this value, the ANN has found a global minimum on the error curve, and oscillations are reasonably well-controlled. Short of using a genetic algorithm to train the ANN to train over a maximally

[23] Although, in the case of *Steghide*, lift is seen when the ANN is trained using only 500 epochs.

effective period of epochs, this is the most direct and intuitive method and yields fair results. During the ANN training period, CPU utilization stayed around 67% - as has been mentioned before, an Artificial Neural Network is a powerful data mining tool, as is evidenced by its resource use and comparatively slow speed, and also its ability to positively identify images amidst only 12 image descriptors (one of which, CLASS, only is used for training the network).

Results of the model training on like members are shown in Appendix E. However, just as important (and much more interesting and potentially fruitful in a zero-knowledge analysis scenario) is the *cross-classification* that may occur *between* data sets – for instance, a suitable question to ask is, *have I managed to train the neural network with enough epochs to make it able to accurately classify new data of the same or similar type while not over-training it?* The answer to this may be found by loading an individual, previously trained, Artificial Neural Network model against data that it has never seen before. Additionally, there will be some "cross-classification" that occurs, say, between *all* static *n*-LSB insertion methods, given the same (or nearly same) end objective. It is certainly not unreasonable to expect that an Artificial Neural Network used in this manner may yield a good degree of accuracy. As such, the following "cross-classifications" were run, and show up in the Appendix E. Lift is seen over initial "auto-predictions" through use of what I will coin "xeno-modelling" – in short, this lends credence to the fact that the models created are *not* overtrained and are *also* flexible enough to pick up *similar* nuances that they have already deduced from their limited training set to make decisions about *new* data. This is exciting, as it demonstrates that in a model built upon *only* 12 image and image histogram attributes, there still exists generalization (of course) *while also providing a fair level of True Positives with respect to a classification of "Infected".*

This last portion is crucial, as this model should strive to maximize True Positives of "Infected" images while reducing false alarms – in effect, when the model reports an alarm, there exists a fairly good chance that there is actually something there. It must also be pointed out that through use of models from *disparate and disjoint types* of n-LSB hiding routines, it is likely possible to produce classification for models, *given that* there exists some overlap of mechanism (or, alternatively, model-perceived mechanism).

4.2. System Uniqueness

The method outlined in this thesis is unique in its simplicity and attempt to find a *minimal* model that is effective across both its own data set, and (ideally) also when examining output from other data sets within its same general method of steganographic encoding. In Appendix E there are several rather dark (15% gray) rows – these rows show prediction classification for *Steghide* using the method outlined in Kurak, as well as the reverse scenario of using methods from Kurak to predict *Steghide*. It would be expected that little overlap would exist between these methods, and the outcome in Appendix E mirrors this assumption. This is appropriate, as the two methods used in this thesis are fundamentally and *mechanically* dissimilar. The methods utilized between the two are different enough in both structure and function such that the use of an Artificial Neural Network is absolutely necessary to even begin to search for possible overlap between the two. However, it also is instructive to note that the model *does not* completely fail – rather, it just provides no additional information about the target that has not already been gathered through the target's own ANN. Given this, it is likely that a model built upon steganographic embedding mechanisms could potentially detect a *similar* type of a steganographic scheme in use – such as may exist between a dynamic insertion model constrained to use 3- and 4-LSB insertions, and the Lie-Chang steganographic

model. This is, in itself, exciting as it opens the theoretical door for finding "primitives" or "roots" of different methods – in effect, a new module would only have to be written for a primitive or root of any given steganographic scheme, and *not* to detect each module itself. This is the first time, in the author's experience, that this notion has been discussed in print as being potentially fruitful. It is likely that this vantage point also runs in parsimony with the notion of finding the smallest model that will yield reasonable results – steganalysis is typically in publication seen as a specialized type of image analysis, and it may be that this view lends itself to honing in on one single, optimized and robust solution that will work for one and only one steganographic method. In short, this is unique in its attempt to lay the groundwork for potentially attacking *families* of steganographic methods, and grouping modules such that *families of families* may be attacked – and recursively continuing in this fashion until, ultimately, it becomes a generalized classification scheme for *all* steganography.

5. CONCLUSION

5.1. Summary

The steganalytic methods tested against in this thesis conform to LSB insertion methods, yet are vastly different in their payload capacity as well as ease of detection. It is to be expected that the method discussed in Kurak's seminal *A Cautionary Note On Image Downgrading*, yielding the highest possible payload capacity of 50%, would also be the easiest to unmask.

This – on the whole – holds true, as the algorithm works by removing the *n* least-significant bits from the cover image and replacing them with the *n most*-significant bits of the embedded image file. Levels employed for *n* in this thesis were 3 and 4, representing the two main levels at which LSB insertion methods are utilized. It is *especially* important, when using this algorithm, to make *careful* selection with regard to a cover image – an image that is "smooth" or otherwise lacking in contextual features is readily defeated through human inspection. Yet, if this constraint is obeyed, it is astonishing how innocuous an infected image may appear – *even when half (4-LSB) or 3/8 (3-LSB) of the image is absent.* The images in Appendix D are demonstrative of this effect – it is important to consider, too, that I chose a *secret* image with strongly-defined borders and edges; this makes it considerably easier for visual detection (but not necessarily so for machine detection).

It is instructive to note that *7.jpg*, having a reasonably smooth and uniform surface (the water), fails to survive use as *both* a 3- and 4-LSB-based stego-image. However, it is also equally important to note that, through intelligent choices in the *context* involved in the carrier files, *12.jpg* and *23.jpg* survive progressively better, with the 3-LSB attacked *23.jpg* file giving

very little insight to the fact that anything is amiss – the image quality isn't the best, but on inspection this could be dismissed as quantization or interpolation artifacts.

Additionally, the second instance of *7.jpg* listed in Appendix D demonstrates the significantly more covert hiding that is achievable via *Steghide*. In this particular file (640x480, without optimal Huffman encoding), the 2KB text file discussed earlier (a portion of the Literature Review of this thesis) was embedded in the image. The initial image was 27.9KB in size, thus yielding a payload size of 7.17% with respect to the initial image size. The resulting image is 28.1KB, and survives visual inspection. The text has been both compressed and encrypted (with a blank pass-phrase), as well as being error-checked by *Steghide*. Using a non-zero-knowledge based approach, and given a known clean (or different) version of the file, it would be possible to examine how the two differed and from this be able to extract information from the stego-image. However, as this is also encrypted, classical cryptography would have to be employed in order to recover the embedded message.

To bring this discussion full-circle, this thesis has demonstrated that a classification system for steganalysis may also take the form of a *minimalist* model in which compact and specialized modules may work in concert to yield fair insight into potentially fruitful image classification, wherein suspect images may then be subject to far more intensive and cumbersome image analysis and feature extraction techniques.

5.2. Discussion of Findings

The findings, although perhaps not as robust as current cutting-edge and special-purpose steganalytic models in publication, suggest that there may indeed be some kernel of interest involved in finding models that are both computationally and intellectually lightweight and straightforward, while at the same time using many small components to examine and

classify images as opposed to components that are both monolithic and cumbersome. It is interesting, too, to note that these results lend themselves rather well to the notion of *steganographic family and/or primitive* discovery and classification. Ideally, given a lightweight model that can more fully represent images while at the same time keeping data volume low, the individual detection and classification modules ought to be able to more fully and richly explore this notion than models currently in publication.

5.3. Limitations of the study & Future Work

This study is an exploratory probe into what will hopefully become a full-scale classification engine for stego-imagery. As such, I have presented two very different instantiations of steganographic embedding using n-LSB insertions. This thesis is intentionally limited in scope, and therefore is constrained analysis of these two methods. Future work will result in the conditions covered in the sections below being achieved.

5.3.1. Pluggable modules

The strength of this hinges upon the ease of writing a series of pluggable modules for analysis and classification. Currently, the implementation of the modules utilized in this thesis is too hands-on and cognitively taxing on an end user. For instance, there is a good deal of hands-on work that is carried out with respect to merging data sets for later analysis through an ANN that should be automated in future revisions of the general model given in this thesis. It is hoped that, in order to avoid software lock-in, a future revision will include the provision of an entire ANN toolbox or module suite, rather than use of any static tool. In short, in future revisions, it would be exciting to allow a user to either "roll their own" ANN software, or go with any open- or closed-source offering and also have the image analysis engine (with its

pluggable modules) automatically feed the ANN software with appropriate data, and track and ideally visualize the results accordingly.

5.3.2. Open system

One main goal of the end-to-end design of the system was that it remain as open and transparent as possible. In its current form, a user is chained to Matlab for image analysis and processing. While this is not a bad move to make (as Matlab is an excellent closed-source tool), it would be more in meeting with the spirit of this thesis to allow for an entirely pluggable tool to work for the end user. For instance, SciLab or even a home-grown image analysis engine would ideally be able to "plug in" to the engine in the future, and provide interoperability between all connected components.

5.3.3. Flexible architecture

The models and methods run in this thesis were housed on a single computer. However, due to the perceived volume and complexity of future applications, it is imperative that future revisions of the engine be open to grid and cluster computing – both PVM and MPI – and ideally include a visualization tool or plug-in module that would enable a user to actively examine the computing nodes and manually assign jobs to each node. The default action, however, ought to be a dynamic and balanced job task being disseminated throughout the grid. Ideally, the hardware used should be transparent, and the system should run identically (albeit possibly magnitudes slower) on a single home machine as on a large grid or supercomputing cluster. There are, of course, problems well beyond the scope of this thesis – or even the end-goal of this project – endemic with grid and cluster-based computing, but the PIRANHA engine should at least strive to make its components and open, flexible, and architecture-independent as possible. Ideally, too, there would exist some sort of a control language that

would be used within the PIRANHA end environment that would then make use of compilers present upon the end system(s) to build and tune appropriate modules as needed.

5.3.4. Hardware Implementation

Whenever working within the confines of software on a complex problem, it is naturally tempting to look to hardware optimization whenever possible. For instance, in a heavily-laden implementation of PIRANHA tasked with viewing a large number of image files (as may be found in a PIRANHA engine running on a system within a network aggregation point), specialized hardware may and should be used to preserve response time. One idea that comes to mind would utilize either PIC or FPGA chip technology, such that an end-user could potentially either write or contract specialized detection and classification modules, and burn these modules to the PIC or FPGA. After which, these modules would be installed in high-density configurations on expansion cards (e.g. PCI) to be placed within a chassis, such that each chassis could have many cards. Then, of course, a rack would ideally be filled with these special-purpose systems, and if the need were great enough, then many racks could be implemented. However, the success of this view hinges upon the constraints of *flexible architecture* and *pluggable modules* being met, as well as the ability to run the engine as a open system.

References

Agency, C. I. (Thu, 05 Sep 2002 16:56:26 GMT). *Minox Camera*, from http://www.cia.gov/cia/information/artifacts/minox.htm

Agency, N. S. (1995). SHA-1.

Anonymous. (1499). *Hypnerotomachia Poliphili: the Dream Battles of Polia's Lover* (1 ed.).

Aura, T. (1995). *Invisible Communication* (Technical Report): Helsinki University of Technology.

Aura, T. (1996). *Practical Invisibility in Digital Communication.* Paper presented at the Information Hiding: First International Workshop.

Avcibas, I., Memon, N. D., & Sankur, B. (2003). Steganalysis Using Image Quality Metrics. *IEEE Transactions on Image Processing, 12*(2), 221-229.

Bacon, F. (1623). *De Augmentis Scientiarum* (Vol. IV).

Bauer, F. L. (1997). *Decrypted Secrets: Methods and Maxims of Cryptology*. Berlin: Springer-Verlag.

Berg, G., Davidson, I., Duan, M.-Y., & Paul, G. (2003, August 12-14). *Searching For Hidden Messages: Automatic Detection of Steganography.* Paper presented at the 15th Innovative Applications of Artificial Intelligence (IAAI) Conference, Acapulco, Mexico.

Bishop, M. (2003). *Computer Security: Art and Science*. Boston. MA: PearsonEducation, Inc.

Brassil, J. T., Low, S. H., Maxemchuk, N. F., & O'Gorman, L. (1995). Electronic Marking and Identification Techniques to Discourage Document Copying. *IEEE Journal of Selected Areas in Communications, 13*(8), 1495-1504.

Buckland, M. K. (2002, November 15-17, 2002). *Histories, Heritages, and the Past: The Case of Emanuel Goldberg.* Paper presented at the Second Conference on the History and Heritage of Scientific and Technical Information Systems, Philadelphia, PA USA.

California, U. o. S.*The USC-SIPI Image Database*, from http://sipi.usc.edu/database/Database.html

Chandramouli, R., Kharrazi, M., & Memon, N. (2003, October 20-22). *Image Steganography and Steganalysis: Concepts and Practice.* Paper presented at the Digital Watermarking, Second International Workshop, IWDW 2003, Seoul, Korea.

Chartrand, S. (2002, January 14, 2002). Using DNA to Detect Counterfeiting. *The New York Times.*

Chen, B., & Wornell, G. (2001). Implementations of Quantization Index Modulation Methods for Digital Watermarking and Information Embedding of Multimedia. *VLSI Signal Processing (Special Issue on Multimedia Signal Processing), 27*, 7-33.

Clelland, C. T., Risca, V., & Bancroft, C. (1999). Hiding Messages in DNA Microdots. *Nature, 399*, 533-534.

Crandall, R. (1998). *Some Notes on Steganography.*Unpublished manuscript.

Crane, R. (1997). *A Simplified Approach to Image Processing: Classical and Modern Techniques in C.* Upper Saddle River, New Jersey 07458: Prentice Hall, Inc.

Dougherty, E. R., & Giardina, C. R. (1987). *Matrix Structured Image Processing*. Englewood Cliffs, New Jersey 07632: Prentice-Hall, Inc.

Eggers, J. J., Bauml, R., & Girod, B. (2002, Jan. 2002). *A Communications Approach to Image Steganography.* Paper presented at the SPIE Electronic Imaging 2002, Security and Watermarking of Multimedia Contents IV, San Jose, USA.

Farid, H. (2002). *Detecting Hidden Messages Using Higher-Order Statistical Models.* Paper presented at the 2002 International Conference on Image Processing, Rochester, New York USA.

Fridrich, J., Goljan, M., & Du, R. (2001). *Steganalysis based on JPEG compatibility.* Paper presented at the Proceedings of SPIE Multimedia Systems and Applications IV, Denver, Colorado USA.

Gold-Software. (2004). Advanced Batch Converter (Version 3.9).

Gray, R. M., Cosman, P. C., & Oehler, K. L. (1993). Incorporating Visual Factors into Vector Quantizers for Image Compression. In A. B. Watson (Ed.), *Digital Images and Human Vision* (pp. 35-52). Cambridge, Mass: Bradford Books.

Herodotus. *The Histories*. London, England: J. M. Dent & Sons, Ltd.

Hetzl, S. (2003). steghide (Version 0.5.1).

Hoover, J. E. (1946, April 1946). The Enemy's Masterpiece of Espionage. *Reader's Digest, 48,* 1-6.

Inc., C. C. (2004). *Popular, Yet Obsolete, Banking Algorithm Broken* (Press Release).

Kahn, D. (1967). *The Codebreakers*. New York, NY: The MacMillan Company.

Kurak, C., & McHugh, J. (1992, 11/30/1992-12/04/1992). *A Cautionary Note On Image Downgrading.* Paper presented at the IEEE Computer Security Applications Conference 1992, San Antonio, TX USA.

Kwan, M. (1998). Snow (Version 1.1).

Kwan, M. (2003). Gifshuffle (Version 2.0).

Leahy, S. (2004, August 10, 2004). Spot-On Solution for Car Thefts. *Wired.com.*

Lee, Y. K., & Chen, L. H. (2000). High Capacity Steganographic Model. *IEEE Proceedings - Vision, Image and Signal Processing, 147*(3), 288-294.

Lie, W.-N., & Chang, L. C. (1999, October 24-28, 1999). *Data Hiding in Images With Adaptive Numbers of Least Significant Bits Based on the Human Visual System.* Paper presented at the International Conference on Image Processing, 1999, Kobe, Japan.

Lin, C.-H., & Lee, T.-C. (1998). A Confused Document Encrypting Scheme and its Implementation. *Computers & Security, 17*(6), 543-551.

Lou, D.-C., & Liu, J.-L. (2002). Steganographic Method for Secure Communications. *Computers & Security, 21*(5), 449-460.

Lou, D.-C., & Liu, J.-L. (submitted). *Common-Cover-Carrier Attack on LSB-Based Steganographic Methods*.Unpublished manuscript.

Low, S. H., Maxemchuk, N. F., Brassil, J. T., & O'Gorman, L. (1995, April 02 - 06, 1995). *Document Marking and Identification Using Both Line and Word Shifting.* Paper presented at the Fourteenth Annual Joint Conference of the IEEE Computer and Communication Societies.

Lyu, S., & Farid, H. (2002, October, 2002). *Detecting Hidden Messages Using Higher-Order Statistics and Support Vector Machines.* Paper presented at the 5th International Workshop on Information Hiding, Noordwijkerhout, The Netherlands.

Machado, R. (1996). EzStego (Version 2.0b4).

Maher, K. (1997). Texto (Version 1.0).

Marshall, D. (2001). *Relationship between DCT and FFT*, from http://www.cs.cf.ac.uk/Dave/Multimedia/node230.html

Marvel, L. M., Hartwig, G. W., & Boncelet, C. G. J. (2000). *Compression-Compatible Fragile and Semi-Fragile Tamper Detection.* Paper presented at the SPIE EI Photonics West, San Jose, CA USA.

Mendoza, C. E. (1999). *Authentication and data integrity of images using the minimax eigenvalue decomposition.* Unpublished M.S., Iowa State University, Ames.

Moroney, C. (1996). Hide and Seek (Version 5.0).

Newman, B. (1940). *Secrets of German Espionage*. London: Robert Hale Ltd.

Newman, R. E., Moskowitz, I. S., Chang, L., & Brahmadesam, M. M. (2002, 2003). *A Steganographic Embedding Undetectable by JPEG Compatibility Steganalysis.* Paper presented at the Information Hiding, 5th International Workshop, IH 2002, Noordwijkerhout, The Netherlands.

Petitcolas, F. A. P., Anderson, R. J., & Kuhn, M. G. (1998, April 14-April 17, 1998). *Attacks on Copyright Marking Systems.* Paper presented at the Proceedings of the Second International Workshop on Information Hiding, Portland, Oregon, USA.

Provos, N. (2004). Stegdetect (Version 0.6).

Rivest, R. (1992). MD5.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning Representations by Back-propagating Errors. *Nature, 323*(9), 533-536.

Schneier, B. (1996). *Applied Cryptography: Protocols, Algorithms, and Source Code in C* (2 ed.): John Wiley & Sons, Inc.

Schneier, B. (2005, February 18). *Schneier on Security*. Retrieved March 12, 2005, from http://www.schneier.com/blog/archives/2005/02/cryptanalysis_o.html

Schott, G. (1680). *Schola Steganographica*: Jobus Hertz.

Shaohui, L., Hongxun, Y., & Wen, G. (2003, July 6-9 2003). *Neural Network Based Steganalysis in Still Images.* Paper presented at the International Conference on Multimedia and Expo, 2003.

Shaohui, L., Hongxun, Y., & Wen, G. (2004, June 15-19 2004). *Steganalysis Based on Wavelet Texture Analysis and Neural Network.* Paper presented at the Fifth World Congress on Intelligent Control and Automation 2004, Hangzhou, China.

Simmons, G. (1984, August 1984). *Prisoners' Problem and the Subliminal Channel.* Paper presented at the CRYPTO '83 - Advances in Cryptology, Santa Barbara, CA USA.

Tacticus, A. (1990). *How to Survive Under Siege*. Oxford, England: Clarendon Press.

Tampa Bay Interactive, I. (1998a, October 25, 2004 19:45:07 EDT). *Baudot Code*. Retrieved March 8, 2005, from http://telecom.tbi.net/baudot.html

Tampa Bay Interactive, I. (1998b, October 25, 2004 19:46:14 EDT). *That Grand Old Teletype*. Retrieved March 8, 2005, from http://telecom.tbi.net/history1.html

The MathWorks, I. (2004). Matlab (Version Student Version Release 14 with Service Pack 1).

Thomas, T. M. (2004). *Network Security First-Step* (May 21, 2004 ed.): Cisco Press.

Trithemius, J. (1621). *Steganographia*. Frankfurt, Germany.

Upham, D. (1999). Jsteg (Version 4.0).

Wallace, G. K. (1991). The JPEG Still Picture Compression Standard. *Communications of the ACM, 34*(4), 30–44.

Wayner, P. (1996). *Disappearing Cryptography*. Chestnut Hill, MA: Academic Press.

Westfeld, A. (2001). *F5 - A Steganographic Algorithm: High Capacity Despite Better Steganalysis.* Paper presented at the Fourth Information Hiding Workshop, Pittsburgh, PA, USA.

Witten, I. H., Frank, E., & Kaufmann, M. (2000). *Data Mining: Practical machine learning tools with Java implementations*. San Francisco.

Yeh, W.-H., & Hwang, J.-J. (2001). Hiding Digital Information Using a Novel System Scheme. *Computers & Security, 20*(6), 533-538.

Zhang, T., & Ping, X. (2003). A New Approach to Reliable Detection of LSB Steganography in Natural Images. *Signal Processing, 83*(10), 2085-2093.

Appendix A – Listing of Steg_nLSB.m

```
% steg_nlsb.m
% Used to n-LSB poison image files.
%     RemoveLSB function from Fabien A. P. Petitcolas
%     (http://www.petitcolas.net/fabien/steganography/image_downgrading/code.html)
% Chris Pilson
% 22 March, 2005
% MS Thesis, Iowa State University
%
% ////////////////////////////////////////////////////////////////////////////

global ENDNUMBER;                          % Number of images to process.
global IMAGEDIRECTORY;                     % Image Directory.
global OUTPUTDIRECTORY;                    % Output Directory.
global Data_to_embed;                      % An image that will be used as hidden data.
global CHANCE_TO_INFECT;
global LSB;                                % How many LSB's to replace?

ENDNUMBER = 747;
LSB=3;
IMAGEDIRECTORY = '320x240_greyscale_75pct_without_optimal_huffman_codes';
OUTPUTDIRECTORY = [IMAGEDIRECTORY,'_Kurak-',int2str(LSB),'LSB'];
if (exist(OUTPUTDIRECTORY)~=7)             % If OUTPUTDIRECTORY doesn't exist...
   mkdir(OUTPUTDIRECTORY);                 % We'll create it.
end
Data_to_embed = imread('hidden.jpg', 'jpeg');
CHANCE_TO_INFECT = 0.33;

InfectedFiles=[];                          % Keep track of infected files.
time=[];


WaitMessage=['Please wait, ',int2str(LSB),'-LSB infecting files... (chance to infect:
',int2str(CHANCE_TO_INFECT*100),'%)'];
h = waitbar(0,WaitMessage);                % Show progressbar.
for i=1:ENDNUMBER
tic;                                       % Start the stopwatch.
   %if(abs(rand) > CHANCE_TO_INFECT)       % Percent chance to infect.
                                           % Damn. Rand doesn't seem to be constrained. :(
   if(bitand
```

```
((abs(rand)>CHANCE_TO_INFECT),(length(InfectedFiles)<(CHANCE_TO_INFECT*ENDNUMBER)))
)
      InfectFile=round(abs(rand*ENDNUMBER));        % Randomly select a file over [0,ENDNUMBER]
      if(InfectFile==0)                             % Make sure we didn't select zero.
         InfectFile=1;
      end
      if(isempty(intersect(InfectedFiles,InfectFile)))% Is this a new infection?
         FileToBeRead=[IMAGEDIRECTORY,'\',int2str(InfectFile),'.jpg'];% Suck in a file.
         STEGnLSB_Image=imread(FileToBeRead);       % Imports the image, equal to "i.JPG".
         Infected = uint8(double(RemoveLSB(STEGnLSB_Image, LSB)) + double(Data_to_embed)
/ 2^(8-LSB));
                                                    % Output an infected file.
         FileToBeWritten=[OUTPUTDIRECTORY,'\',int2str(InfectFile),'.jpg'];
         InfectedFiles=[InfectedFiles, InfectFile];
         imwrite(Infected,FileToBeWritten);         % Write out the infected file.
      end
   end
waitbar(i/ENDNUMBER);
time = [time,toc];                                  % How many seconds did this pass take (vector)?
end
close(h);                                           % Close the waitbar handler (window).

%     Write out CSV Files

OutFileName=[OUTPUTDIRECTORY,'\infected.csv'];
ColumnNames=['FILE,',int2str(LSB),'LSB_INFECTED'];
dlmwrite(OutFileName, ColumnNames, '');             % Write a header file for the CSV that'll come
out of this showing infected files.
csvwrite(OutFileName,rot90(InfectedFiles),1,0);

%     Now do some time reporting...
TimeDataName=['Steg_nLSB_',OUTPUTDIRECTORY,'_runtimes.csv'];
ColumnNames=['Min,Mean,Max,StDev'];
dlmwrite(TimeDataName, ColumnNames, '');
ColumnValues=[min(time);mean(time);max(time);std(time)];
dlmwrite(TimeDataName, rot90(ColumnValues), '-append', 'newline', 'pc');
```

Appendix B – Listing of Steghide.m

```
% Steghide.m
% Used to steghide-poison image files.
%
% Chris Pilson
% 22 March, 2005
% MS Thesis, Iowa State University
%
% ////////////////////////////////////////////////////////////////////////////

global ENDNUMBER;                          % Number of images to process.
global IMAGEDIRECTORY;                     % Image Directory.
global OUTPUTDIRECTORY;                    % Output Directory.
global Data_to_embed;                      % A file that will be used as hidden data.
global CHANCE_TO_INFECT;
global LSB;                                % How many LSB's to replace? Changed @ runtime.

ENDNUMBER = 747;
STEGHIDECOMMAND = 'tools\steghide\steghide.exe';
IMAGEDIRECTORY = '640x480_greyscale_75pct_without_optimal_huffman_codes';
OUTPUTDIRECTORY = ['Steghide','.',IMAGEDIRECTORY];
if (exist(OUTPUTDIRECTORY)~=7)             % If OUTPUTDIRECTORY doesn't exist...
   mkdir(OUTPUTDIRECTORY);
end
Data_to_embed = 'PilsonLitReview.txt';
CHANCE_TO_INFECT = 0.33;

Batchfile=[];                              % Create a batch file to run.
InfectedFiles=[];                          % Keep track of infected files.
time=[];


WaitMessage=['Please wait, StegHide infecting files... (chance to infect:
',int2str(CHANCE_TO_INFECT*100),'%)'];
h = waitbar(0,WaitMessage);                % Show progressbar.
for i=1:ENDNUMBER
tic;                                       % Start the stopwatch.
   %if(abs(rand) > CHANCE_TO_INFECT)        % Percent chance to infect.
                                           % Damn.  Rand doesn't seem to be constrained. :(
   if(bitand
```

```
((abs(rand)>CHANCE_TO_INFECT),(length(InfectedFiles)<(CHANCE_TO_INFECT*ENDNUMBER)))
)
      InfectFile=round(abs(rand*ENDNUMBER));        % Randomly select a file over [0,ENDNUMBER]
      if(InfectFile==0)                             % Make sure we didn't select zero.
        InfectFile=1;
      end
      if(isempty(intersect(InfectedFiles,InfectFile)))% Is this a new infection?
        FileToBeRead=[IMAGEDIRECTORY,'\',int2str(InfectFile),'.jpg'];% Suck in a file.
        %Steghide_Image=imread(FileToBeRead);     % Imports the image, equal to "i.JPG".
        BatchfileLine=[STEGHIDECOMMAND,'  embed  -cf  "',FileToBeRead,'"  -ef
',Data_to_embed,' -sf "',OUTPUTDIRECTORY,'\',int2str(InfectFile),'.jpg"',' -p """];
        [status, result] = system(BatchfileLine);  % Execute steghide on the chosen file.
        if (status == 0)                            % Fail silently; do nothing if status != 0.
          InfectedFiles=[InfectedFiles, InfectFile]; % Go ahead and mark this as a "successful"
infection.
        end
      end
    end
waitbar(i/ENDNUMBER);
time = [time,toc];                                  % How many seconds did this pass take (vector)?
end
close(h);                                           % Close the waitbar handler (window).

%     Write out CSV Files

OutFileName=[OUTPUTDIRECTORY,'\infected.csv'];
ColumnNames=['FILE,StegHide_INFECTED'];
dlmwrite(OutFileName, ColumnNames, '');             % Write a header file for the CSV that'll
come out of this showing infected files.
csvwrite(OutFileName,rot90(InfectedFiles),1,0);

TimeDataName=['Steghide_',OUTPUTDIRECTORY,'_runtimes.csv'];
ColumnNames=['Min,Mean,Max,StDev'];
dlmwrite(TimeDataName, ColumnNames, '');
ColumnValues=[min(time);mean(time);max(time);std(time)];
dlmwrite(TimeDataName, rot90(ColumnValues), '-append', 'newline', 'pc');
```

Appendix C – Listing of ANNPreprocess.m

```
% ANNPreprocess.m
% Used to preprocess JPEG images for running through an ANN.
% Chris Pilson
% 22 March, 2005
% MS Thesis, Iowa State University
%
% ////////////////////////////////////////////////////////////////////////////

global ENDNUMBER;                          % Number of images to process.
global IMAGEDIRECTORY;                     % Image Directory.
global OUTPUTDIRECTORY;                    % Where to dump resultsets.
Copy1stOrderCommand = ['copy hist_headers.txt+'];
Copy2ndOrderCommand = ['copy '];
ENDNUMBER = 747;
if (exist(OUTPUTDIRECTORY)==7)             % Does the OUTPUTDIRECTORY exist from a
prior run?
   IMAGEDIRECTORY = OUTPUTDIRECTORY;       % If so, then we'll use it as the INPUT
directory for this run.
end
OUTPUTDIRECTORY=[IMAGEDIRECTORY,'.','ANN_out'];    % Set the new OUTPUT directory.
if (exist(OUTPUTDIRECTORY)~=7)             % If the new OUTPUTDIRECTORY doesn't exist...
   mkdir(OUTPUTDIRECTORY);                 % We'll create it.
end
dlmfile=[OUTPUTDIRECTORY,'\hist_headers.txt'];
dlmwrite(dlmfile,   'HistMAX,HistMIN,HistSTDEV,HistVARIANCE,HistCLASS,HistBugFixForWEKA',
'');
dlmfile=[OUTPUTDIRECTORY,'\bitplane_headers.txt'];
dlmwrite(dlmfile, 'Entropy,Mean,StDev,VMax,VMin,VMean,StDev2,CLASS,BugFixForWEKA', '');

InfectedCSVFile = [IMAGEDIRECTORY,'\infected.csv'];
InfectedFiles = csvread(InfectedCSVFile);
InfectedFiles = sort(InfectedFiles);

bitplane_stats=[];
histogram_stats=[];

time=[];

WaitMessage=['Please   wait,   Processing   Images   for   ANN...   (target   directory:
```

```
',OUTPUTDIRECTORY,')'];
h = waitbar(0,WaitMessage);                    % Show progressbar.
for i=1:ENDNUMBER
tic;                                  % Start the stopwatch.
   if (ismember(i,InfectedFiles))              % Is the current image a member of the infected files?
      CLASS=1;                          % If yes, we'll classify it as "+", for "infected".
   else
      CLASS=0;                          % Otherwise, not infected.
   end
   FileToBeRead=[IMAGEDIRECTORY,'\',int2str(i),'.jpg'];
   CPImage_gray=imread(FileToBeRead);          % Imports the image, equal to "i.JPG".

   % ///////////////// FIRST-ORDER STATISTICS ///////////////
   % Getting the BITPLANE:
   CPImage_double = double(CPImage_gray);              % Make a DOUBLE representation of
CPImage_gray
   SaveToBitplane=[OUTPUTDIRECTORY,'\bitplane_',int2str(i),'.txt']; % Build the filename to save
under.
   CPImage_diff = diff(CPImage_gray);          % Get n-1 differences within CPImage_gray.
   SaveBitplaneDiffs=[OUTPUTDIRECTORY,'\bitplane_diff_',int2str(i),'.txt'];
   %csvwrite(SaveBitplaneDiffs,CPImage_diff);
   CPImage_entropy = entropy(CPImage_gray);
   CPImage_double_mean = mean2(CPImage_double);    % Get max, min, mean values (nnn2 for
matrix).
   CPImage_double_stdev = std2(CPImage_double);    % Get Standard Deviation for image
COLUMNS (stdev2 for matrix).
   CPImage_double_vmax = var(max(CPImage_double));  % Variance of max values in columns.
   CPImage_double_vmin = var(min(CPImage_double));  % Variance of min values in columns.
   CPImage_double_vmean = var(mean(CPImage_double));
   CPImage_double_vstdev = var(std(CPImage_double));

CPImage_double_stats=[CPImage_entropy,CPImage_double_mean,CPImage_double_stdev,CPI
mage_double_vmax,CPImage_double_vmin,CPImage_double_vmean,CPImage_double_vstdev,
CLASS,0];
bitplane_stats=[bitplane_stats;CPImage_double_stats];

   % Working with the HISTOGRAM:
   CPImage_imhist = imhist(CPImage_gray);          % Stores the histogram to CPImage_imhist. Y
variable is the vector position [0,255]
   SaveImhist=[OUTPUTDIRECTORY,'\Histogram_',int2str(i),'.txt'];
   CPImage_imhist_diff = diff(CPImage_imhist);     % Get n-1 differences within CPImage_imhist.
   SaveImhistDiff=[OUTPUTDIRECTORY,'\Histogram_diff_',int2str(i),'.txt'];
```

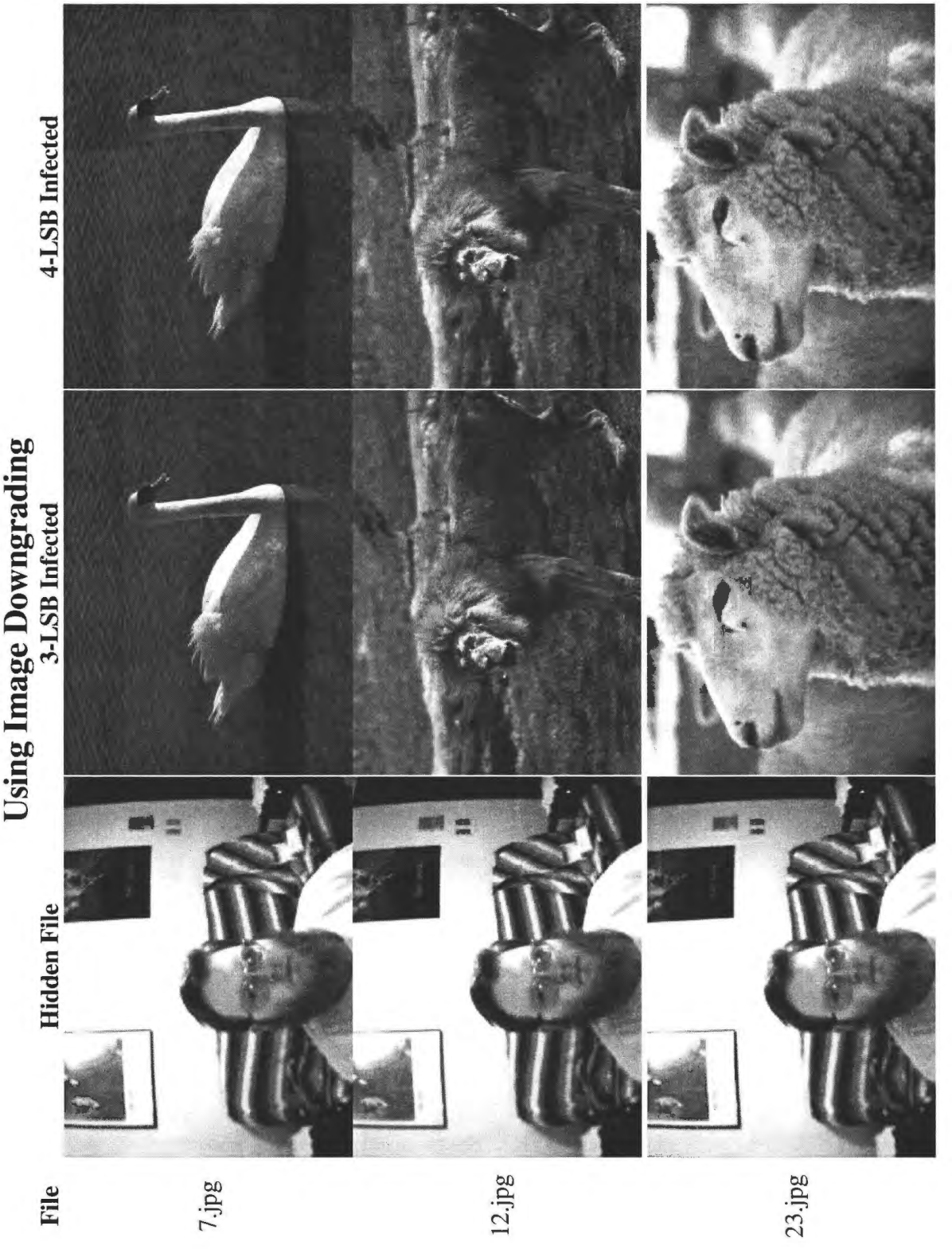
```
   CPImage_imhist_max = max(CPImage_imhist);
   CPImage_imhist_min = min(CPImage_imhist);
   CPImage_imhist_stdev = std(CPImage_imhist);        % Get Standard Deviation for histogram.
   CPImage_imhist_var = var(double(CPImage_imhist));    % Get variability of image histogram.
Store as string.

CPImage_imhist_dstats=[CPImage_imhist_max,CPImage_imhist_min,CPImage_imhist_stdev,CP
Image_imhist_var,CLASS,0];
histogram_stats=[histogram_stats;CPImage_imhist_dstats];
SaveImhistDStats=[OUTPUTDIRECTORY,'\Histogram_dstsats_',int2str(i),'.txt'];
waitbar(i/ENDNUMBER);
time = [time,toc];                          % How many seconds did this pass take (vector)?
end

outfile=[OUTPUTDIRECTORY,'\',IMAGEDIRECTORY,'.bitplane_stats.csv'];
csvwrite(outfile,bitplane_stats);
outfile=[OUTPUTDIRECTORY,'\',IMAGEDIRECTORY,'.histogram_stats.csv'];
csvwrite(outfile,histogram_stats);
close(h);                                   % Kill off the waitbar handle.

%     Now do some time reporting...
TimeDataName=['ANNPreProcess_',OUTPUTDIRECTORY,'_runtimes.csv'];
ColumnNames=['Min,Mean,Max,StDev'];
dlmwrite(TimeDataName, ColumnNames, '');
ColumnValues=[min(time);mean(time);max(time);std(time)];
dlmwrite(TimeDataName, rot90(ColumnValues), '-append', 'newline', 'pc');
```

Appendix D – Visual Inspection of Infected Files



Using *Steghide*

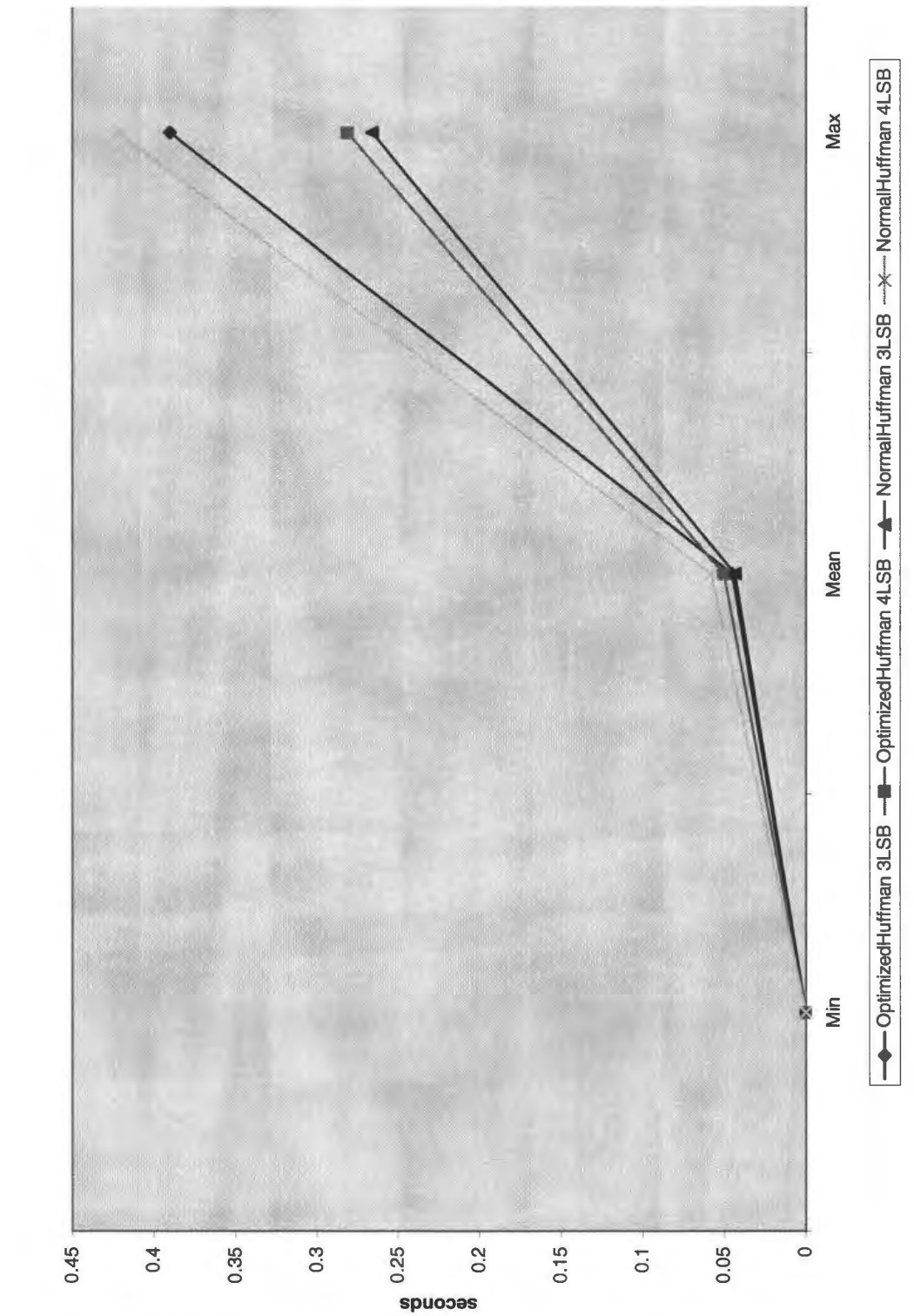
File	Hidden File	3-LSB Infected	4-LSB Infected
7.jpg	Text file – 2KB from the *Literature Review* section of this thesis.		

Appendix E – Artificial Neural Network Classification on Auto- and Xeno-Data

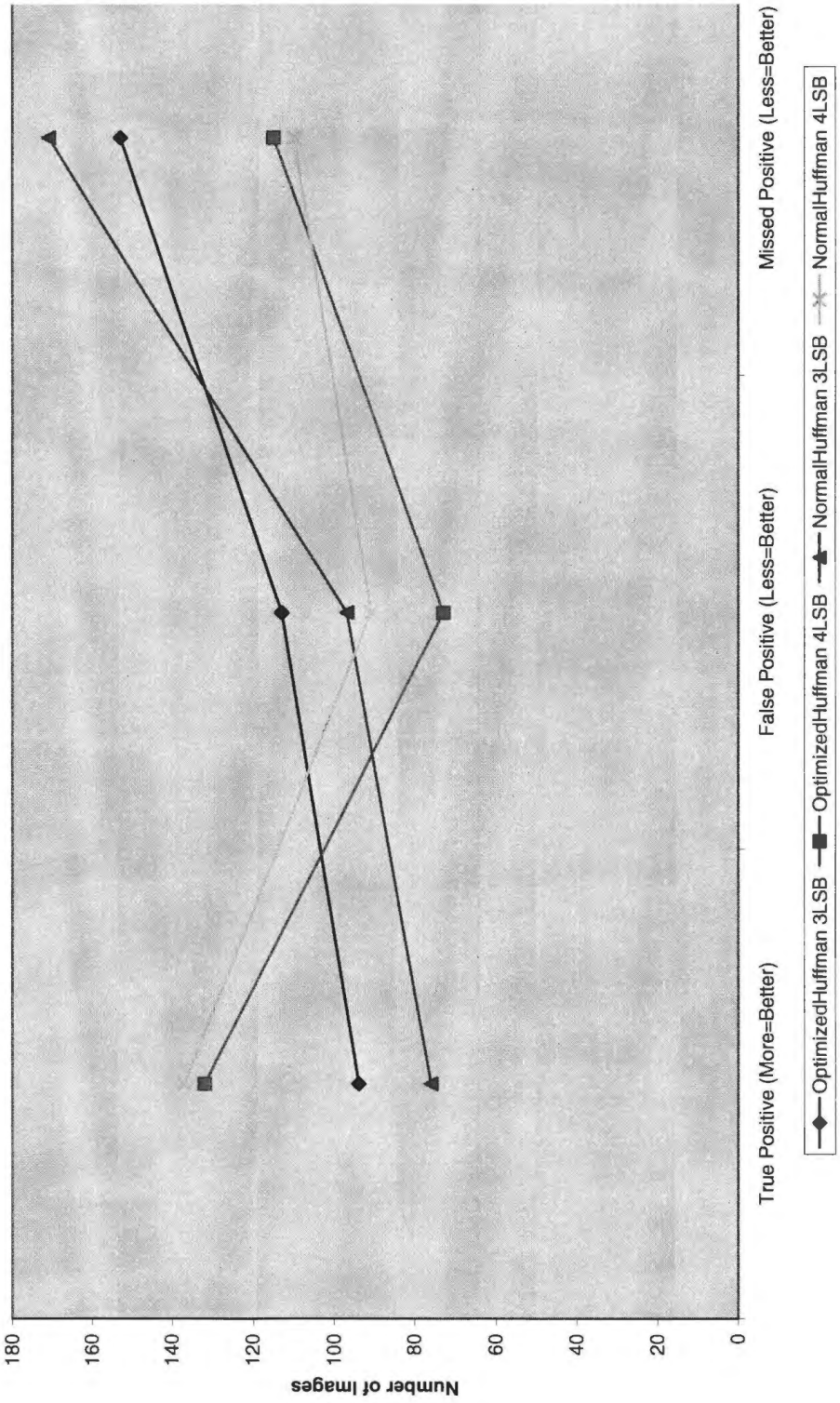
Model and application	TP Rate Infected	FP Rate Infected	TP Rate Clean	FP Rate Clean	Accuracy (Percent)	Gain over SELF?
320x240_greyscale_75pct_optimal_huffman_codes_Kurak-3LSB	0.381	0.226	0.774	0.619	64.3909	
320x240_greyscale_75pct_optimal_huffman_codes_Kurak-4LSB	0.534	0.146	0.854	0.466	74.8327	
320x240_greyscale_75pct_without_optimal_huffman_codes_Kurak-3LSB	0.308	0.194	0.806	0.692	64.1232	
320x240_greyscale_75pct_without_optimal_huffman_codes_Kurak-4LSB	0.555	0.182	0.818	0.445	73.0924	
Steghide.640x480_greyscale_75pct_with_optimal_huffman_codes.merged_stats (trained on 500 epochs only)	0.101	0.084	0.916	0.899	64.6586	
Steghide.640x480_greyscale_75pct_with_optimal_huffman_codes.merged_stats	0.126	0.12	0.88	0.874	63.0522	
Steghide.640x480_greyscale_75pct_without_optimal_huffman_codes.merged_stats	0.105	0.114	0.886	0.895	62.7845	
320x240_greyscale_75pct_optimal_huffman_codes_Kurak-3LSB.merged_stats						
→ 320x240_greyscale_75pct_optimal_huffman_codes_Kurak-4LSB.merged_stats	0.721	0.106	0.721	0.106	83.668	X
→ 320x240_greyscale_75pct_without_optimal_huffman_codes_Kurak-3LSB.merged_stats	0.316	0.13	0.316	0.13	68.6747	X
→ 320x240_greyscale_75pct_without_optimal_huffman_codes_Kurak-4LSB.merged_stats	0.676	0.094	0.906	0.324	82.9987	X
→ Steghide.640x480_greyscale_75pct_with_optimal_huffman_codes.merged_stats	0.194	0.204	0.796	0.806	59.7055	No
→ Steghide.640x480_greyscale_75pct_without_optimal_huffman_codes.merged_stats	0.17	0.216	0.784	0.83	58.0991	No
320x240_greyscale_75pct_optimal_huffman_codes_Kurak-4LSB.merged_stats						
→ 320x240_greyscale_75pct_optimal_huffman_codes_Kurak-3LSB.merged_stats	0.401	0.12	0.88	0.599	72.1553	X
→ 320x240_greyscale_75pct_without_optimal_huffman_codes_Kurak-3LSB.merged_stats	0.316	0.13	0.87	0.684	68.6747	X
→ 320x240_greyscale_75pct_without_optimal_huffman_codes_Kurak-4LSB.merged_stats	0.676	0.094	0.906	0.324	82.9987	X
→ Steghide.640x480_greyscale_75pct_with_optimal_huffman_codes.merged_stats	0.194	0.204	0.796	0.806	59.7055	No
→ Steghide.640x480_greyscale_75pct_without_optimal_huffman_codes.merged_stats	0	0	1	1	66.9344	No
320x240_greyscale_75pct_without_optimal_huffman_codes_Kurak-3LSB.merged_stats						
→ 320x240_greyscale_75pct_optimal_huffman_codes_Kurak-3LSB.merged_stats	0.401	0.12	0.88	0.599	72.1553	X
→ 320x240_greyscale_75pct_optimal_huffman_codes_Kurak-4LSB.merged_stats	0.571	0.116	0.884	0.429	78.0455	X
→ 320x240_greyscale_75pct_without_optimal_huffman_codes_Kurak-4LSB.merged_stats	0.676	0.094	0.906	0.324	82.9987	X
→ Steghide.640x480_greyscale_75pct_with_optimal_huffman_codes.merged_stats	0.016	0.016	0.984	0.984	66.3989	X
→ Steghide.640x480_greyscale_75pct_without_optimal_huffman_codes.merged_stats	0	0	1	1	66.9344	No

Model and application	TP Rate Infected	FP Rate Infected	TP Rate Clean	FP Rate Clean	Accuracy (Percent)	Gain over SELF?
320x240_greyscale_75pct_without_optimal_huffman_codes_Kurak-4LSB.merged_stats						
→ 320x240_greyscale_75pct_optimal_huffman_codes_Kurak-3LSB.merged_stats	0.401	0.12	0.88	0.599	72.1553	X
→ 320x240_greyscale_75pct_optimal_huffman_codes_Kurak-4LSB.merged_stats	0.571	0.116	0.884	0.429	78.0455	X
→ 320x240_greyscale_75pct_without_optimal_huffman_codes_Kurak-3LSB.merged_stats	0.316	0.13	0.87	0.684	68.6747	X
→ Steghide.640x480_greyscale_75pct_with_optimal_huffman_codes.merged_stats	0.016	0.016	0.984	0.984	66.3989	X
→Steghide.640x480_greyscale_75pct_without_optimal_huffman_codes.merged_stats	0.032	0.008	0.992	0.968	67.4699	X

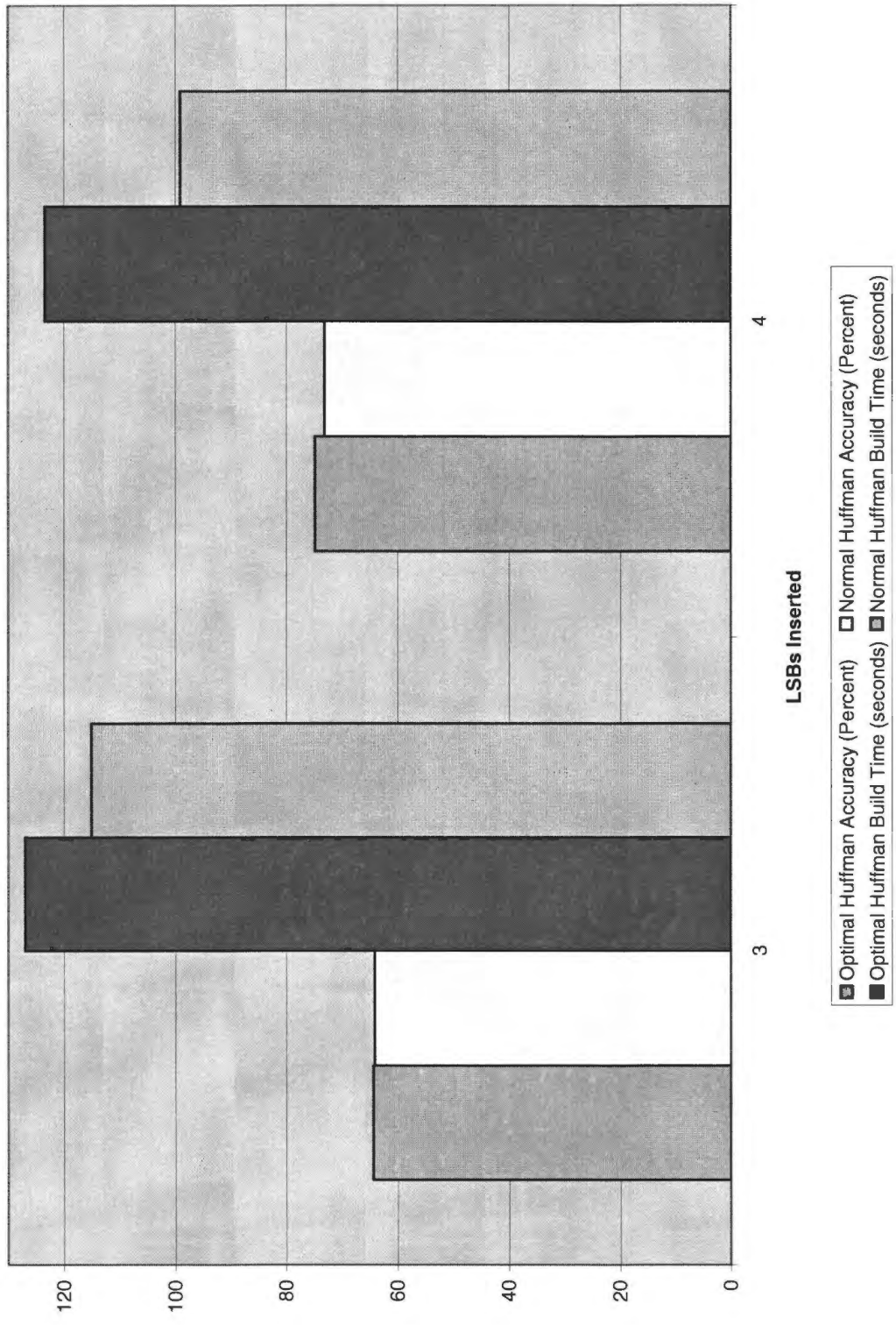
Appendix F – Insertion Runtime vs. Type of Image Downgrade



Appendix G – True, False, and Missed Positives Detected



Appendix H – System Build Time and Accuracy vs. LSB Insertion



Appendix I – Grayscale Photo Plates for Images Utilized


ENTRANCE


CLARE COLLEGE


IMPASSABLE
WHEN WET


FRIED DOUGH